Eduardo Mendel do Nascimento

Investigação Experimental usando Algoritmos Populacionais em Ambientes Ruidosos

Vitória - ES, Brasil

01 de Julho de 2011

Eduardo Mendel do Nascimento

Investigação Experimental usando Algoritmos Populacionais em Ambientes Ruidosos

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo para obtenção do título de Mestre em Informática.

Orientador:
Thomas Walter Rauber

Programa de Pós-Graduação em Informática
Departamento de Informática
Centro Tecnológico
Universidade Federal do Espírito Santo

Vitória - ES, Brasil

01 de Julho de 2011

Dissertação de Mestrado sob o título *"Investigação Experimental usando Algoritmos Populacionais em Ambientes Ruidosos"*, defendida por Eduardo Mendel do Nascimento e aprovada em 01 de Julho de 2011, em Vitória, Estado do Espírito Santo, pela banca examinadora constituída pelos professores:

Prof. Ph.D. Thomas Walter Rauber
Orientador

Prof. Dr.Sc. Flávio Miguel Varejão
Examinador Interno

Profa. D.Sc. Isabel C. M. Rosseti
Examinadora Externa

"Without music, life would be a mistake."

Friedrich Nietzsche

Sumário

Siglas

BBPSO Bare Bones Particle Swarm.

BBPSO+C_dJ BBPSO com salto segundo distribuição de probabilidade de Cauchy.

BBPSO+CJ BBPSO com estratégia de salto caótico.

BBPSO+CJGM BBPSO com salto caótico usando mapa de Gauss.

BBPSO+CJLM BBPSO com salto caótico usando mapa logístico.

BBPSO+CJZM BBPSO com salto caótico usando mapa Zaslaswkii.

BBPSO+G_dJ BBPSO com salto segundo distribuição de probabilidade Gaussiana.

EA Algoritmo Evolutivo.

FIPS Fully Informed Particle Swarm.

FIPS+CJ FIPS com estratégia de salto caótico.

GA Algoritmo Genético.

PSO Particle Swarm Optimization.

PSOgbest PSO com topologia global.

PSOgbest+CJ PSOgbest com estratégia de salto caótico.

PSOlbest PSO com topologia local em anel.

PSOlbest+CJ PSOlbest com estratégia de salto caótico.

SI Inteligência Coletiva.

Lista de Figuras

Lista de Tabelas

1 *Introdução*

"Come join me now I know the way,
(We'll be free!)
No king to rule us down
(Ride the sea!)
Man the boats and hoist the sails,
(All hands on deck!)
There's land to find I know,
(Don't look back!)."

Eric The Red
Rebellion

O objetivo principal deste trabalho é a realização de um estudo investigativo experimental de algoritmos populacionais em ambientes ruidosos. Quatro versões do algoritmo *Particle Swarm Optimization* (PSO) foram adotadas nos experimentos. São elas: PSO canônico (topologia global), PSO padrão (topologia local), *Bare Bones* (BBPSO) e *Fully Informed Particle Swarm* (FIPS). O PSO é um algoritmo populacional muito poderoso e com grande potencial para solução de problemas de otimização. Esse método consiste na otimização de uma função objetivo por meio da troca de informações entre indivíduos (partículas) de uma população (enxame). Por meio de cooperação, as abordagens de otimização baseadas em população frequentemente encontram soluções satisfatórias. Entretanto, a maioria das versões desenvolvidas nos últimos anos apresenta dificuldades na otimização de funções com muitos mínimos locais em espaços de alta dimensão. Foi analisado o comportamento de uma estratégia denominada de estratégia de *jump* em ambientes incertos com a finalidade de estudar melhorias no desempenho do métodos investigados, tanto no contexto estático quanto ruidoso.

Inicialmente, os algoritmos populacionais foram investigados com a estratégia de *jump* com objetivo de escapar de mínimos locais. O *jump* é utilizado quando não são

observadas melhorias durante o processo de otimização. Essa abordagem foi apresentada primeiramente com base nas distribuições de probabilidade Gaussiana e de Cauchy. Testes experimentais foram conduzidos em um conjunto bem conhecido de problemas multimodais, com muitos mínimos locais. Os resultados obtidos sugerem que as versões híbridas dos algoritmos populacionais são capazes de superar o desempenho de suas respectivas versões originais. Conclui-se que a estratégia de *jump* é bastante eficaz no combate à convergência prematura. A melhora no desempenho é consequência de um pequeno, mas importante, número de saltos bem sucedidos.

Em virtude dos resultados promissores obtidos com os algoritmos populacionais combinados com a estratégia de *jump*, essa abordagem foi expandida e, ao invés de gerar números randômicos segundo distribuições de probabilidade, optou-se por utilizar sequências caóticas. As quatro versões do PSO foram novamente investigadas com objetivo de analisar a habilidade da estratégia modificada em permitir que indivíduos em regime de estagnação escapem de atratores sub-ótimos. Resultados de simulação demonstram que a adição de saltos caóticos impulsionam o desempenho dos algoritmos em comparação com as abordagens utilizando distribuição de probabilidade. Além disso, a estratégia populacional com caos possui custo computacional inferior.

Na sequência do trabalho, o foco é a investigação da introdução de *jump* com caos nos algoritmos populacionais aplicados, entretanto, a problemas de otimização ruidosos, com adição de incerteza às funções de teste. As versões FIPS e BBPSO ainda não haviam sido aplicadas anteriormente na literatura a funções ruidosas. O método híbrido é analisado experimentalmente em diversas funções *benchmarks*. Resultados de simulação indicam que a adição da estratégia de *jump* é benéfica em termos de robustez. Em ambientes ruidosos, um algoritmo para ser considerado robusto precisa alcançar sistematicamente uma solução satisfatória. Não é suficiente obter bons resultados sob baixos níveis de ruídos e degradar a solução à proporção que o nível de ruído aumenta. Conclui-se que embora o desempenho dos algoritmos populacionais deteriore com o aumento no nível de ruído, as soluções encontradas pelas versões modificadas são muito superiores às soluções obtidas pelas versões originais. A estratégia de saltos adicionada aos algoritmos populacionais aplicada tanto em funções estáticas quanto em funções ruidosas demonstrou ser bastante eficiente e eficaz pois aumenta a chance do algoritmo escapar de mínimos locais. A abordagem analisada é simples e de fácil implementação, sem nenhum acréscimo de esforço computacional. O uso de sequências caóticas em substituição às distribuições de probabilidade

contribui para a eficiência da abordagem. Além disso, é importante notar que essa estratégia pode ser expandida para outras variantes do PSO, bem como outros algoritmos evolutivos.

2 *Inteligência Coletiva*

"We don't come alone
We are fire we are stone
We're the hand that writes
Then quickly moves away
We search for the truth
We could die upon the tooth
But the thrill of just the chase
Is worth the pain"

The Last in Line
Dio

Um bando de pássaros que sobrevoam o céu, um grupo de formigas que buscam por alimentos, um cardume de peixes que nadam, mudam de curso e fogem juntos são alguns exemplos de comportamentos coletivos existentes na natureza (SHAW, 1962). A variedade dos comportamentos de animais e insetos desperta o interesse de muitos pesquisadores e é alvo de muitos estudos. Biólogos, cientistas da computação, engenheiros e psicólogos têm estudado o comportamento social de populações e enxames com intuito de compreender melhor como esses animais se interagem, como alcançam seus objetivos e como evoluem. Inteligência Coletiva (SI, do inglês *Swarm Intelligence*) é a denominação da subárea da Inteligência Computacional que abrange um conjunto de metodologias e técnicas inspiradas no comportamento coletivo inteligente observado em bandos de pássaros, cardumes, enxames e colônias de formigas.

A expressão SI foi introduzida pela primeira vez por Beni e Wang em 1989, no contexto de sistemas robóticos, e extendida posteriormente por (BONABEAU; DORIGO; THERAULAZ, 1999; EBERHART; SHI; KENNEDY, 2001). Essa área de pesquisa enfatiza o comportamento social e cooperativo de organismos que vivem e interagem em grupos de indivíduos autônomos e simples. Muitos aspectos das atividades coletivas

de organismos sociais são auto-organizáveis, significando que um comportamento complexo emerge a partir de interações entre indivíduos não sofisticados. Os indivíduos (ou agentes) utilizam regras relativamente simples para governar suas ações.

De um modo geral, *Swarm Intelligence* reúne uma gama de algoritmos aplicados principalmente a problemas de otimização e de planejamento. Esses algoritmos simulam a dinâmica de populações de pequenos indivíduos que se comunicam entre si, desenvolvendo uma inteligência coletiva que os permite solucionar problemas com sofisticado grau de complexidade e de maneira extremamente eficiente. Como exemplos de problemas solucionados com a utilização dessas técnicas, pode-se citar: treinamento de rede neural artificial (BERGH; ENGELBRECHT, 2002); otimização multi objetiva (HU; EBERHART, 2002); despacho econômico de energia elétrica (VICTOIRE; JEYAKUMAR, 2004); e controle descentralizado de veículos autônomos (BARAS; TAN; HOVARESHTI, 2004).

2.1 Inspiração na Natureza

O enorme fascínio exercido pelo comportamento de animais na natureza sobre pesquisadores na comunidade científica, principalmente na área de inteligência computacional, é justificado pela complexidade na descrição, modelagem e simulação do movimento orquestrado exibido por diversas populações de animais. Colônias de insetos, por exemplo, parecem trabalhar de maneira coordenada, ainda que não haja um único membro no controle. Cupins constroem estruturas gigantescas, formigas localizam alimentos com muita eficiência, e bandos de pássaros e cardumes de peixes movem-se como um só organismo na defesa contra predadores.

Há duas categorias principais no quesito comportamento social: insetos sociais que vivem em colônias pois seus membros são incapazes de sobreviverem sozinhos, e espécies cujos indivíduos se organizam coletivamente para se auto beneficiarem.

2.1.1 Insetos Sociais

Apesar da relativa simplicidade individual de cada inseto, uma colônia é capaz de construir estruturas muito complexas. Os montes construídos por cupins, por exemplo, possuem aberturas para ventilação e cultivo de fungos para nutrição. Colônias de formigas também apresentam comportamento coletivo igualmente complexo ao examinarem eficientemente diversas regiões em busca de alimentos, mesmo estes estando distribuídos ou dispersos em áreas distintas.

A metodologia de simulação do comportamento de formigas na busca por alimentos ilustra perfeitamente a ideia presente nas técnicas de inteligência coletiva. Em 1990, pesquisadores liderados por Jean Louis Deneubourg realizaram um clássico experimento em que ficou demonstrado que, dado dois caminhos diferentes conectando o ninho à comida, uma colônia de formigas escolhe, com alta probabilidade, o caminho mais curto. Deneubourg et al. mostraram que esse comportamento pode ser explicado via um modelo probabilístico simples em que cada formiga decide o caminho a seguir randomicamente com base na intensidade de feromônio existente no solo; quanto maior a concentração de feromônio maiores as chances das formigas seguirem esse caminho. O feromônio é depositado pelas próprias formigas durante o percurso. O intuito é guiar outras formigas em direção ao alimento. A concentração de feromônio é mais intensa nas regiões visitadas com maior frequência em virtude da distância percorrida pelas formigas para chegarem à comida e retornarem

ao ninho; em geral, a concentração de feromônio é superior nas proximidades do ninho. É esse efeito que viabiliza a localização do caminho mais curto. A evaporação do feromônio previne estagnação e evita convergência prematura em caminhos não ótimos. A Figura 2.1 ilustra o processo de descoberta do caminho mais curto entre o ninho e a comida, segundo o esquema descrito por Deneubourg et al..

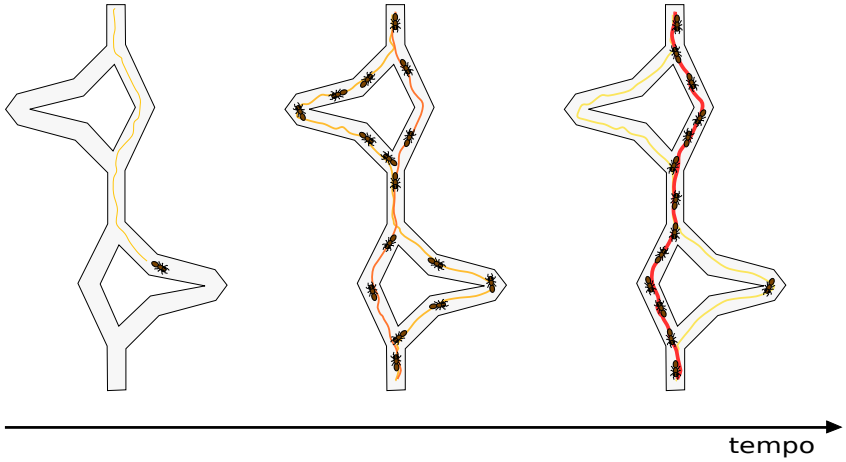


Figura 2.1: Ilustração gráfica de um experimento com formigas para descoberta do caminho mais curto.

2.1.2 Bandos, Cardumes e Manadas

Bandos, cardumes e manadas podem ser constituídos por milhares de indivíduos. Cada um desses indivíduos são limitados em suas capacidades mentais e de percepção. Pode-se assumir que apenas algumas poucas regras simples (e locais) controlam o movimento de um único animal. O aspecto local das regras indica que são considerados somente membros de uma determinada vizinhança (dependendo da percepção de cada indivíduo). A regra básica observada na formação de grupos é evitar colisões ao mesmo tempo em que se mantém a proximidade entre os indivíduos.

Cada espécie de animal possui um nível diferente de percepção. O resultado dessa diversidade é uma vasta gama de possíveis comportamentos coletivos distintos. Os peixes, por exemplo, não enxergam tão longe como os pássaros, especialmente em águas turvas, mas conseguem, usando um órgão sensorial denominado de linha lateral, detectar pequenas variações na água em seu entorno ocasionadas pela movimentação de peixes vizinhos (PITCHER; PARTRIDGE; WARDLE, 1976). A visão de longo alcance dos pássaros, por outro lado, permite-lhes visualizar a movimentação de indivíduos a distâncias enormes. Essa característica possibilita ao bando se preparar para uma rápida mudança de direção; altas velocidades de propagação das

"ondas de manobra" normalmente não são explicadas estritamente apenas por regras locais e pelo tempo de reação das aves (POTTS, 1984).

Em 1987, Reynolds criou um modelo computacional de movimento coordenado de animais denominado de modelo de animóides (*boids*, em inglês). Esse modelo básico para descrição do comportamento de indivíduos em um enxame consiste de três conceitos simples baseados em posição e velocidade. São eles: anticolisão; alinhamento, para alinhar os indivíduos segundo a direção de movimento dos animóides vizinhos; e coesão, para aproximá-los caso se afastem do bando. A ideia desses conceitos está representada na Figura 2.2.

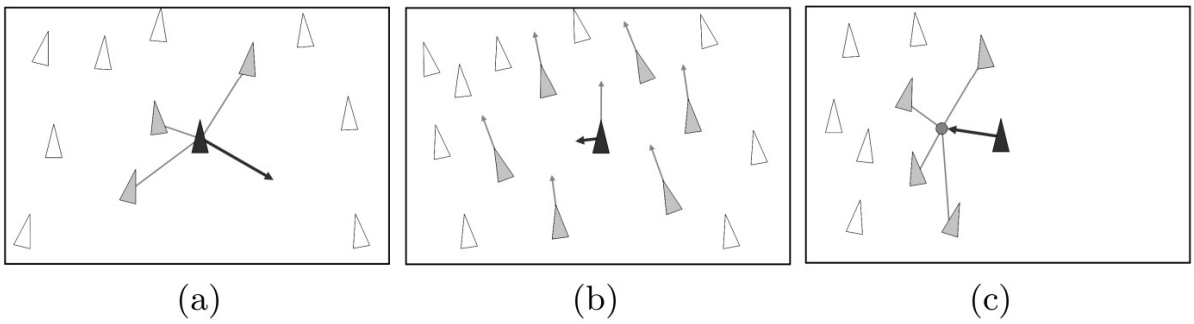


Figura 2.2: Conceitos para simulação de animóides. (a) Anticolisão. (b) Alinhamento. (c) Coesão.

2.2 Meta heurísticas

Os algoritmos baseados na metodologia SI são constituídos de indivíduos que representam soluções candidatas ao problema. As buscas efetuadas por esses algoritmos populacionais são guiadas pela natureza social dos indivíduos. Os membros de uma população se comunicam uns com os outros com intuito de direcionar as buscas, emulando o comportamento de indivíduos em uma sociedade. O objetivo não é imitar fielmente o fenômeno social intrínseco à comunidade, mas sim utilizar alguns de seus aspectos em aplicações práticas.

A ausência, originalmente, de um mecanismo de seleção nos algoritmos populacionais é o que os difere dos algoritmos evolucionários. Apesar de haver, em geral, na estratégia de inteligência coletiva, atualizações das soluções candidatas encontradas no decorrer da busca, a identidade de cada indivíduo é constante ao longo do tempo (KENNEDY, 2003). Pode-se afirmar que os indivíduos se aperfeiçoam com o tempo; o que não é válido para sistemas evolutivos. Neste último existe competição entre os

indivíduos e seus descendentes a cada iteração.

Os métodos de otimização mais conhecidos na área de inteligência coletiva são: otimização por enxame de partículas (do inglês *Particle Swarm Optimization*, abreviado para PSO) (EBERHART; KENNEDY, 1995), otimização por colônia de formigas (ACO, do inglês *Ant Colony Optimization*) (DORIGO; CARO; GAMBARDELLA, 1999), sistema imune artificial (CASTRO; TIMMIS, 2002), busca bacteriana por alimentos (PASSINO, 2002) e colônias de aranhas (BOURJOT; CHEVRIER; THOMAS, 2003). Existem diversas outras abordagens.

O método computacional ACO foi desenvolvido em analogia ao fenômeno biológico exibido pelas sociedades de formigas na busca por alimentos (DORIGO; CARO; GAMBARDELLA, 1999; DORIGO; BIRATTARI; STÜTZLE, 2006). Esse método baseia-se em trilhas artificiais de feromônio como meio de comunicação entre agentes simples de uma colônia. Trilhas de feromônios são usadas por formigas como informações numéricas distribuídas na construção de soluções com base em probabilidades. O primeiro algoritmo segundo essa abordagem, denominado de sistema de formiga (AS, do inglês *Ant System*), foi introduzido por Dorigo, Maniezzo e Colorni (1991). Diversas melhorias foram propostas em trabalhos posteriores.

O algoritmo ACO tem sido aplicado com sucesso a vários problemas de otimização combinatorial (DORIGO; GAMBARDELLA, 2002; CARO; DORIGO, 1998; KORB; STÜTZLE; EXNER, 2006). Bonabeau, Dorigo e Theraulaz (1999) argumentaram que esse algoritmo é aplicável a muitos problemas de otimização desde que a construção incremental das soluções possa, de alguma maneira, ser representada por um grafo. As formigas podem atravessar o grafo com base em uma regra de transição probabilística e uma simples heurística descrevendo a vontade de cada uma delas.

Outro algoritmo populacional bastante conhecido é o Enxame de Partículas (PSO). O PSO é um método de otimização estocástica baseado em população de indivíduos inspirada no comportamento social de bandos de pássaros e cardumes de peixes. Foi proposto inicialmente por Eberhart e Kennedy em 1995. Esse método será descrito com maiores detalhes no capítulo a seguir. Variações dessa abordagem também foram investigadas e serão explicadas adiante.

3 *Enxame de Partículas*

"I will take the train to anywhere
Go through heaven and through hell
I will make my way and I don't care
If I win or fail
What will come and what will be
No one can foresee
Future gives the answers to all questions
Some may choose a simple life
Some decide to die
Some are always searching for the reasons"

The Circle Of Life
Freedom Call

O método de otimização por enxame de partículas é uma técnica de computação estocástica baseada em dinâmica de populações (KENNEDY; EBERHART, 1995). Esse método consiste na otimização de uma função objetivo por meio da troca de informações entre indivíduos (partículas) de uma população (enxame). Através de cooperação, as abordagens de otimização baseadas em população frequentemente encontram soluções satisfatórias.

No algoritmo PSO, a solução de um problema de otimização global é representada por um ponto (partícula) ou uma superfície em um espaço n-dimensinal. As hipóteses são lançadas em um espaço com uma velocidade inicial e um canal de comunicação entre as partículas. As partículas se movimentam pelo espaço de solução e são avaliadas segundo algum critério objetivo específico para cada problema. Ao longo do tempo, as partículas são aceleradas em direção às regiões mais promissoras, isto é, aquelas com os melhores valores calculados pela função objetivo.

Nos últimos anos, PSO tem sido usado em diversas áreas de pesquisa e aplicado

a vários problemas com muito sucesso. Demonstrou-se empiricamente que o desempenho do algoritmo *particle swarm* pode superar em termos de eficácia da solução e custo de execução computacional outros métodos de otimização bem estabelecidos na literatura. Outra característica que justifica o uso do PSO e o torna bastante atrativo são os pouco parâmetros de ajuste necessários. Uma única versão pode funcionar satisfatoriamente em uma grande variedade de aplicações.

3.1 Comportamento Social

O fenômeno social subjacente ao pensamento humano é muito mais complexo que o comportamento coreografado de bandos, enxames e cardumes de outras espécies. Essencialmente a dimensão do espaço do pensamento é muito superior às três dimensões convencionais. Além disso, nesse espaço cognitivo não existem colisões; a convergência de duas ou mais mentes é denominada consenso. Entretanto, de certo modo, o algoritmo *particle swarm* procura simular determinadas características da sociedade humana. Sempre que as pessoas concordam entre si, elas "caminham" em direção a um mesmo ponto no espaço de crenças, tornando-se mais semelhantes na medida em que influenciam e imitam umas as outras. Padrões e culturas emanam dessas relações. Quando discordam, a distância entre si, nesse espaço de crenças e opiniões, aumenta.

A dinâmica do *particle swarm* é baseada em analogias de interações sociais. Dado um problema, uma população de partículas, análoga a uma população de seres humanos, discute possíveis soluções para o problema. A interação entre indivíduos viabiliza o compartilhamento de soluções promissoras (KENNEDY, 2008). Uma descrição interessante a respeito da lógica subjacente à ideia que originou o método PSO é apresentada no trabalho de Eberhart, Shi e Kennedy (2001). Nesse livro, os autores discutem inclusive alguns aspectos de filosofia social presentes na abordagem.

No PSO original, cada membro da população é um pouco professor e aprendiz, em um esquema de cego guiando cego. Regiões promissoras do espaço de busca, onde boas soluções foram encontradas, são favorecidas. A população (ou enxame) somente é eficaz quando seus indivíduos são capazes de receber e usar informações adquiridas por outros membros, e de transmitir os resultados de suas experiências aos seus vizinhos. A colaboração é o aspecto mais importante no processo de busca da metodologia *particle swarm*. Na ausência de comunicação, as partículas perdem a habilidade de se aprimorarem na busca por melhores soluções de um problema.

3.2 Estrutura e Trajetória das Partículas

Cada partícula i no algoritmo PSO é composta essencialmente por três vetores, todos com a mesma dimensionalidade do espaço do problema: posição $\boldsymbol{x}_i$; melhor posição previamente descoberta, $\boldsymbol{p}_{best_i}$; e velocidade $\boldsymbol{v}_i$. Ao todo, cinco componentes

são usados na descrição da trajetória de uma partícula:

- $\boldsymbol{x}_i = [x_{i1}, x_{i2}, \cdots, x_{in}]^\mathsf{T}$, a posição da partícula i no espaço de busca de n dimensões. Esse vetor representa o conjunto de valores da hipótese corrente. A dimensão de $\boldsymbol{x}_i$ é dada pelo número de variáveis do problema a ser otimizado.

- $f(\boldsymbol{x})$ que significa a qualidade da solução representada pelo vetor $\boldsymbol{x}$, calculada de acordo com a função objetivo (*fitness*, em inglês) específica do problema. Essa função pode ser bastante simples, como $(x_1 + x_2)^2$, ou muito complexa, contendo diversas restrições.

- $\boldsymbol{v}_i = [v_{i1}, v_{i2}, \cdots, v_{in}]^\mathsf{T}$ é o vetor velocidade associado à partícula i. Esse vetor representa a variação no deslocamento de uma partícula a cada novo passo. Alterações na velocidade implicam em alterações na direção do movimento de uma partícula pelo espaço de busca.

- $\boldsymbol{p}_{best_i}$ contém a melhor solução já encontrada pela partícula i. Basicamente $\boldsymbol{p}_{best_i}$ é uma cópia do vetor $\boldsymbol{x}_i$ no ponto em que o indivíduo i identificou a melhor solução até determinado momento de sua busca. Em conjunto, $\boldsymbol{p}_{best_i}$ e $\boldsymbol{x}_i$ corrente compõem a "memória" da partícula, usada para orientar a partícula pelo espaço de regiões promissoras.

- $f(\boldsymbol{p}_{best_i})$ é um valor numérico associado à melhor solução descoberta, segundo uma função objetivo.

Todo membro da população também tem conhecimento da melhor partícula, ou seja, aquela com o melhor valor calculado pela função objetivo, dentre todos os membros de sua vizinhança. As vizinhanças podem ser compostas por pequenos grupos de partículas. Na alternativa global, todas as partículas da população constituem uma única vizinhança. Nesse caso, o melhor indivíduo é denominado $\boldsymbol{g}_{best}$.

A trajetória de uma partícula é computada pela adição do seu vetor velocidade, $\boldsymbol{v}$, ao seu vetor deslocamento, $\boldsymbol{x}$, obtendo uma nova posição no espaço de soluções. As investigações conduzidas em regiões promissoras durante a busca por melhores soluções é diretamente influenciada pelo processo no qual o vetor $\boldsymbol{v}$ é modificado. Ainda, o PSO introduz um ajuste aleatório a essa influência para variar o tamanho do passo. Essa variação reduz as chances de uma partícula "cair em buracos" e seguir o mesmo caminho indefinidamente. Matematicamente, a trajetória das partículas é descrita pelas equações a seguir (KENNEDY; EBERHART, 1995):

$$\boldsymbol{v}_i(t+1) = \boldsymbol{v}_i(t) + \rho_1 r_1 \cdot (\boldsymbol{p}_{best_i} - \boldsymbol{x}_i) + \rho_2 r_2 \cdot (\boldsymbol{g}_{best} - \boldsymbol{x}_i) \tag{3.1}$$

$$\boldsymbol{x}_i(t+1) = \boldsymbol{x}_i(t) + \boldsymbol{v}_i \tag{3.2}$$

em que:

- ρ_1 and ρ_2 são constantes positivas representando a taxa de aprendizado cognitivo e a taxa de aprendizado social, respectivamente; e
- r_1, $r_2 \sim U(0,1)$ são números randômicos gerados a partir de uma distribuição de probabilidade no intervalo [0;1].

3.3 Topologia Populacional

O estabelecimento de uma noção virtual de topologia permite que as partículas do algoritmo sejam influenciadas socialmente por apenas uma pequena parcela da sua população, os seus vizinhos virtuais. Entende-se por topologia populacional (ou sociometria) uma rede social que define a relação existente entre indivíduos de uma mesma vizinhança (MENDES; NEVES, 2004). A influência social rege o movimento de cada indivíduo, isto é, os indivíduos são guiados por informações encontradas em suas respectivas vizinhanças. O objetivo fundamental dessa ideia é combater a convergência prematura do algoritmo e, consequentemente, melhorar seu desempenho e sua robustez. A redução da quantidade de fontes de influências possibilita que uma determinada partícula, ao encontrar uma solução considerada boa, influencie diretamente apenas seus próprios vizinhos.

Os primeiros trabalhos sobre PSO (EBERHART; KENNEDY, 1995; KENNEDY; EBERHART, 1995) adotavam uma única estrutura topológica, chamada *gbest*. Nessa topologia, a influência social exercida sobre cada partícula tinha sua origem na partícula mais bem sucedida dentre todas da população. Isso equivale a uma rede social em que todos os indivíduos estão conectados a todos os outros. Todas as partículas pertencem à mesma vizinhança global, ou seja, cada uma é capaz de determinar o valor da função objetivo das demais e decidir qual é a "melhor". O algoritmo PSO padrão utiliza sempre a melhor informação disponível. Tipicamente, os indivíduos ajustam suas próprias trajetórias com base nas soluções encontradas pelos seus vizinhos topológicos. Cada partícula realiza buscas pelo espaço em direção à melhor

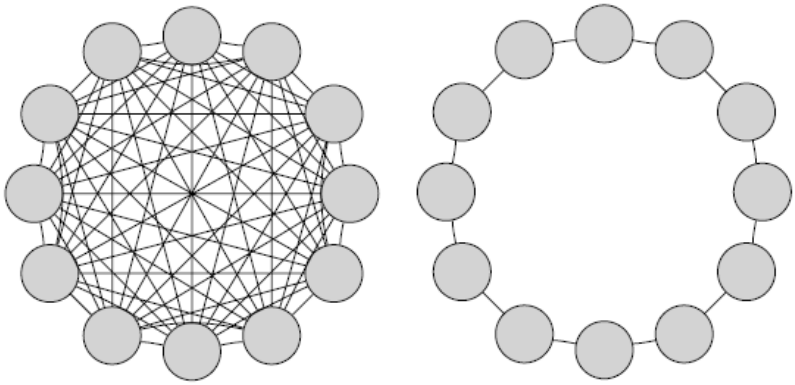
região encontrada por qualquer membro de sua vizinhança (influência social) e também em direção à região mais promissora encontrada anteriormente por ela mesma (aprendizado por experiência). Essas informações são usadas na definição do centro de oscilação dessa partícula. Quando todas as partículas da população fazem parte de uma mesma vizinhança, o desempenho do algoritmo é deteriorado. A topologia global permite que soluções ruins sejam propagadas muito rapidamente (KENNEDY; MENDES, 2006). Há uma tendência em orientar mal a população em direção a um mínimo local prematuramente. Entretanto, enquanto as partículas exploram o espaço de busca, elas eventualmente podem localizar soluções melhores (MENDES; KENNEDY; NEVES, 2003). Um bom questionamento está relacionado com a importância de centrar a trajetória de uma partícula entorno de um centro realmente bom. Embora em alguns casos seja vantajoso usar a melhor posição disponível no espaço, em outros um centro modesto pode ser mais eficaz.

A estrutura topológica populacional afeta a taxa em que uma informação é disseminada pela população. Segundo alguns esquemas alternativos à organização populacional global vigente no algoritmo PSO original, o tempo necessário para que informações disponibilizadas por uma partícula alcançem todas as demais é maior visto que uma informação é compartilhada inicialmente apenas entre os indivíduos da uma mesma vizinhança.

A habilidade do *particle swarm* em explorar o espaço de busca pode ser controlado por meio de sua topologia: populações densamente conectadas apresentam elevada velocidade de convergência e grande potencial para intensificação (exploração local); enquanto populações com conexões esparsas exploram o espaço para descoberta de novas áreas promissoras ainda não exploradas (KENNEDY; MENDES, 2002; KENNEDY, 1999), diversificando as buscas.

Várias estruturas topológicas foram propostas nos últimos anos como alternativa ao modelo global. Dentre elas, destaca-se a topologia local ou modelo *lbest*. Resultados experimentais reportam melhorias no desempenho do PSO com o uso da estrutura topológica local em problemas de otimização multimodais (BRATTON; KENNEDY, 2007). A topologia local, inicialmente proposta por (KENNEDY, 1999), é diametralmente oposta à topologia global. Nela, todos os indivíduos estão esparsamente conectados e a informação é propagada em um ritmo demasiadamente lento. Essa topologia foi proposta como uma maneira mais apropriada para lidar com problemas complexos, superando diversas dificuldades existentes no modelo global tradicional.

Na estrutura sociométrica *lbest*, a população forma conceitualmente um anel de indivíduos. Cada indivíduo é conectado a n partículas imediatamente à sua direita e n partículas imediatamente à sua esquerda. Um indivíduo influencia e é influenciado apenas pelos seus vizinhos próximos. Observa-se empiricamente que vizinhanças locais reduzem a velocidade de propagação das informações. O objetivo é evitar convergência prematura. A Figura 3.1 ilustra as duas topologias mais estudadas na área de *particle swarm*, o modelo *gbest* e o modelo *lbest*.



(a) Modelo global (b) Modelo local em anel

Figura 3.1: Topologias do PSO.

A topologia *lbest* possibilita que subpopulações explorem regiões diversas do espaço simultaneamente, pois partes da população, localizadas espacialmente em regiões distantes umas das outras, são independentes. Um segmento da população pode convergir em direção a um ótimo local, enquanto outro segmento converge para um ótimo diferente ou continua a sua busca. As informações são propagadas vizinho a vizinho. Eventualmente todas as partículas serão direcionadas a um único ponto. O fluxo de informação de uma subpopulação para outra é moderado pela necessidade de "persuasão" de indivíduos intermediários. Sempre que uma partícula descobre uma região promissora no espaço de busca, ela influencia diretamente somente seus vizinhos. Seus vizinhos de segundo grau são influenciados após aqueles diretamente conectados à partícula inicial se tornarem altamente bem sucedidos. A influência de um indivíduo "viaja" lentamente por outras vizinhanças.

Topologia Dinâmica Vizinhança dinâmica permite que partículas façam parte de diferentes grupos em momentos distintos (semelhante aos enxames na natureza) e ajustem suas "opiniões" a respeito da melhor área no espaço. Incentivar a troca de informações entre uma partícula e subconjuntos distintos (e randômicos) de partículas

a cada passo, pode possibilitar a investigação de diferentes direções de busca no espaço, ocasionando um melhor desempenho do algoritmo. Muitos trabalhos já investigaram a inclusão de dinamismo na topologia do algoritmo *particle swarm*. Suganthan sugeriu, em 1999, alterar o foco do PSO, de diversificação para intensificação, aumentando gradualmente o número de vizinhos em cada partícula até que todas as partículas sejam incluídas na mesma vizinhança. Mohais et al. (2005) usaram vizinhança randômica no PSO, em conjunto com operadores dinâmicos para modificação da estrutura da vizinhança: migração aleatória de vizinho e reestruturação completa da topologia. Na primeira abordagem, uma partícula cuja vizinhança contém um ou mais indivíduos é selecionada aleatoriamente; um de seus vizinhos é removido e reintroduzido na vizinhança de outra partícula também escolhida ao acaso, mas com a vizinhança ainda incompleta. Na segunda abordagem, a estrutura da vizinhança é mantida constante durante um determinado número de iterações e, então, completamente reinicializada. Em (AKAT; GAZI, 2008) os autores analisaram o desempenho do algoritmo PSO segundo estratégias baseadas tanto no conceito de distância entre partículas para delimitação de áreas de percepção, quanto na ideia de especificação randômica de vizinhos a cada iteração do algoritmo.

3.4 Otimização por Enxame de Partículas Canônico

Em sua forma mais simples, as partículas do algoritmo PSO calculam, a cada passo, novas soluções para o problema, isto é, novas posições no espaço de busca, como combinação entre as melhores posições individuais de cada partícula e a melhor posição visitada por quaisquer outro indivíduo da população, com a adição de pequenas pertubações aleatórias. O algoritmo canônico consiste basicamente na atualização das velocidades e das posições de cada partícula da população. O objetivo é direcioná-las às regiões mais promissoras do espaço.

3.4.1 Tamanho da População

A quantidade de indivíduos de uma população é parâmetro específico de cada problema. Em geral, varia entre 20 e 50 partículas. O algoritmo *particle swarm* necessita de um número menor de indivíduos que a maioria das estratégias evolutivas para alcançar soluções de alta qualidade. Carlisle e Dozier (2001) variaram o tamanho da população, de 5 até 200, e realizaram comparações de acordo com a taxa de sucesso,

média de iterações e média de avaliações. Eles concluíram que uma população com 30 indivíduos é uma ótima escolha; pequena o suficiente para o algoritmo manter-se eficiente, e grande o suficiente para produzir resultados satisfatórios.

O número de partículas em cada vizinhança, segundo a estrutura topológica adotada, também é uma parâmetro considerado específico de cada problema e que influencia bastante a convergência do algoritmo. Em (EBERHART; KENNEDY, 1995), conjecturou-se que uma vizinhança local com poucas partículas é ideal para evitar mínimos locais, enquanto uma vizinhança global (composta por todos os indivíduos) apresenta uma velocidade de convergência maior. Suganthan (1999) conduziu diversos experimentos em que o tamanho da vizinhança de cada partícula era aumentado gradativamente, até incorporar todos os indivíduos da população, porém seus resultados – ainda que interessantes – foram inconclusivos.

3.4.2 Velocidade Máxima

Na versão original do algoritmo PSO, as velocidades das partículas eram limitadas a um valor máximo, v_{max} (KENNEDY, 2005b). Quando a velocidade excedia esse valor v_{max}, em qualquer coordenada, seu valor era truncado para o valor máximo permitido.

O parâmetro v_{max} era importante para o algoritmo. Se o seu valor fosse muito alto, as partículas "voariam" além de boas soluções. Por outro lado, um valor de v_{max} muito baixo ocasionaria uma exploração excessivamente lenta, inviabilizando a localização de boas soluções. As partículas seriam presas fáceis para mínimos locais, incapazes de se moverem para fora de bacias de atração.

3.4.3 Controle dos Limites

A maioria dos trabalhos na área de *particle swarm* limita as buscas do algoritmo a um hipercubo. É importante forçar, de algum modo, que a exploração seja conduzida sempre no interior de um hiperespaço válido. Em geral, as partículas são reposicionadas dentro de limites conhecidos sempre que necessário. Em outras abordagens, entretanto, sugere-se que uma alternativa interessante seria simplesmente não realizar qualquer tipo de restrição quanto aos limites da busca. A função objetivo – *fitness* – poderia ser modificada para atribuir o valor $+\infty$ (assumindo um problema de minimização) para soluções não válidas; as partículas voltariam prontamente para dentro

do espaço válido. Todavia, há indícios experimentais que desse modo as partículas perdem bastante tempo vagando por espaços não válidos.

3.4.4 Ponderação de Inércia

Notou-se que os efeitos do limite máximo para a velocidade, v_{max}, e a sua arbitrariedade eram insatisfatórios para diversos problemas; as partículas falhavam ao convergir. O uso de v_{max} impedia as oscilações das partículas de diversificação (exploração global) para intensificação (exploração local).

Em pouco tempo o conceito de ponderação de inércia (ou peso inercial) foi estabelecido (SHI; EBERHART, 1998a; SHI; EBERHART, 1998b). O objetivo era controlar com maior eficiência o alcance da busca, reduzindo, ou até mesmo eliminando, a importância do v_{max}. A equação (3.3) demonstra a fórmula de velocidade do PSO modificada pela inclusão do coeficiente de inércia α.

$$\boldsymbol{v}_i(t+1) = \alpha \boldsymbol{v}_i(t) + \boldsymbol{U}[0,\varphi_1] \cdot (\boldsymbol{p}_{best_i} - \boldsymbol{x}_i) + \boldsymbol{U}[0,\varphi_2] \cdot (\boldsymbol{g}_{best} - \boldsymbol{x}_i) \tag{3.3}$$

Experimentos sugerem que com o uso de ponderação inercial, o fator v_{max} para limitação da velocidade pode simplesmente ser ajustado para o valor do intervalo de cada variável (em cada dimensão). Essa limitação é por vezes necessária para evitar que partículas permaneçam oscilando em alta velocidade ao redor de uma região promissora sem explorá-la adequadamente. Dessa maneira, não há mais necessidade de definir uma estratégia para ajuste do parâmetro v_{max}.

3.4.5 Fator de Constrição

Fator de Constrição foi introduzido por Clerc e Kennedy (2002) baseado na análise matemática da dinâmica do PSO. O objetivo dessa análise é especificar intervalos de bons valores para os parâmetros do algoritmo, garantindo convergência local.

Inicialmente, o parâmetro v_{max} era usado para limitar a velocidade. Kennedy mostrou que existem diversas maneiras para se definir um sistema PSO genérico de modo a controlar a convergência do algoritmo. Um sistema simples foi descrito por Clerc e Kennedy (2002):

$$\boldsymbol{v}_i = \lambda \left[\boldsymbol{v}_i + \boldsymbol{U}[0, \varphi_1] \cdot (\boldsymbol{p}_{best_i} - \boldsymbol{x}_i) + \boldsymbol{U}[0, \varphi_2] \cdot (\boldsymbol{g}_{best} - \boldsymbol{x}_i) \right] \tag{3.4}$$

Clerc e Kennedy introduziram um coeficiente de constrição λ para garantir a convergência:

$$\lambda = \frac{2k}{|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|}, \quad \text{em que } k \in [0,1], \varphi = \varphi_1 + \varphi_2, \varphi > 4 \tag{3.5}$$

Em geral, quando o fator de constrição é usado, φ é ajustado para 4.1 ($\varphi_1 = \varphi_2 = 2.05$) e $k = 1$, determinando um coeficiente $\lambda \approx 0.729$. Isso é equivalente ao modelo com ponderação inercial em que $\alpha \approx 0.729$ e $\varphi_1 = \varphi_2 \approx 1.49445$. Quando $k = 1$, a convergência é lenta o suficiente para assegurar exploração antes que a busca convirja. Certamente, existem outros valores para os coeficientes de constrição. Uma análise detalhada a respeito da derivação desse fator pode ser encontrada em (CLERC; KENNEDY, 2002).

A principal vantagem desse algoritmo é o uso de uma média ponderada estocástica com as informações disponíveis, ou seja, as influências cognitiva ($\boldsymbol{p}_{best_i}$) e social ($\boldsymbol{g}_{best}$). Quando esses dois pontos estão próximos, a partícula oscila ao redor da região definida por eles, e eventualmente converge para o centro dessa região. Em contrapartida, quando estão distantes, a trajetória da partícula é mais ampla, priorizando a exploração de uma área mais extensa entre essas duas posições.

Outro aspecto interessante acontece quando os membros de uma vizinhança se agrupam na mesma região. Nesse caso, as partículas começam a explorar intensamente essa área. Se um membro da vizinhança descobre outro ótimo, as trajetórias das partículas são expandidas novamente para examinarem essa nova região. De fato, essa partícula irá influenciar outras para também explorarem a mesma região recém descoberta.

A vantagem do uso do coeficiente de constrição é que não é necessário usar o parâmetro v_{max} ou conjecturar valores para qualquer parâmetro de controle de convergência. Experimentos subsequentes (EBERHART; SHI, 2002) indicaram que é interessante igualar v_{max} à x_{max}, o limite de cada variável em cada dimensão. O resultado é um *particle swarm* sem parâmetros específicos de problemas, considerado o algoritmo PSO canônico.

3.4.6 Atualização Síncrona ou Assíncrona

Na versão mais comum do algoritmo PSO, a estrutura topológica da população pode ser analisada como uma lista circular encadeada. Considera-se a vizinhança local de uma partícula como uma janela deslizante ao longo dos seus dois lados. Portanto, é natural que o $\boldsymbol{l}_{best}$ seja calculado de maneira assíncrona, isto é, logo que uma partícula inicia sua trajetória o seu melhor vizinho é determinado e essa influência transmitida ao deslocamento da próxima partícula. Por outro lado, no caso global, é mais natural realizar a avaliação do $\boldsymbol{g}_{best}$ de modo síncrono, entre as iterações, paralelizando o movimento das partículas.

A principal diferença entre as duas abordagens está relacionada ao momento de identificação da partícula mais bem sucedida. Na atualização síncrona, todos as partículas movem-se em paralelo antes do melhor indivíduo ser selecionado. Entretanto, na atualização assíncrona, enquanto os vizinhos de um dos lados da partícula em ajuste já foram atualizados, os membros do lado oposto ainda não.

Carlisle (2002) realizou diversos experimentos, executando o PSO tanto de maneira assíncrona quanto síncrona, para investigar se a ordem na atualização das partículas influencia no resultado final do algoritmo. Carlisle concluiu que, em geral, atualizar a população de modo assíncrono influencia positivamente a taxa de convergência do PSO.

3.4.7 O Algoritmo Canônico

O método consiste na aplicação repetidas vezes das equações de atualização da velocidade e posição, respectivamente descritas nas fórmulas (3.4) e (3.2). Um pseudocódigo básico dessa abordagem é apresentado no Algoritmo 1. As inicializações mencionadas nos primeiros passos do algoritmo consistem das seguintes etapas:

1. Inicializar cada coordenada $\boldsymbol{x}_i$ com um valor randômico extraído de uma distribuição uniforme dentro do intervalo $[\ \underline{x}\ ,\ \overline{x}\ ]$, $\forall\ i \in 1,\ldots,P$. Todas as partículas são distribuídas uniformemente pelo espaço de busca.

2. As velocidades são inicializadas com valor 0, uma vez que as posições iniciais são aleatórias. Como alternativa, pode-se inicializar as velocidades de cada coordenada, $v_{i,j}$, segundo uma distribuição uniforme dentro de um intervalo definido para todo $i \in 1,\ldots,P$ e $j \in 1,\ldots,d$.

3. Inicializar $\boldsymbol{p}_{best_i} = \boldsymbol{x}_i$, $\forall\, i \in 1,\ldots,P$.

Em geral, o algoritmo é executado durante um número fixo pré-estabelecido de iterações (ou avaliações da função objetivo, *fitness*), ou até que um erro limite especificado seja encontrado.

Algoritmo 1 PSO Canônico (topologia global)

Parâmetros de entrada: Tamanho P da população.
// inicialização aleatória da população de partículas usando distribuição de probabilidade
// uniforme, em que $\underline{x}$ e $\overline{x}$ são os limites inferior e superior respectivamente.
para cada partícula i **faça**
$\boldsymbol{x}_i = \underline{x} + (\overline{x} - \underline{x}) \cdot U(0,1)$
$\boldsymbol{p}_{best_i} = \boldsymbol{x}_i$
fim para
$\boldsymbol{g}_{best} = \arg\min_{i=1,\ldots,P}\{f(\boldsymbol{x}_i)\}$ // melhor indivíduo da população
repetir
para cada partícula i **faça**
para cada dimensão j da partícula i **faça** // atualizar $\boldsymbol{x}_i$ segundo (3.2) e (3.4)
$\boldsymbol{v}_i = \lambda\left[\boldsymbol{v}_i + \boldsymbol{U}[0,\varphi_1]\cdot(\boldsymbol{p}_{best_i} - \boldsymbol{x}_i) + \boldsymbol{U}[0,\varphi_2]\cdot(\boldsymbol{g}_{best} - \boldsymbol{x}_i)\right]$
$\boldsymbol{x}_i = \boldsymbol{x}_i + \boldsymbol{v}_i$
fim para
se $f(\boldsymbol{x}_i) < f(\boldsymbol{p}_{best_i})$ **então** // atualizar melhor posição da partícula
$\boldsymbol{p}_{best_i} = \boldsymbol{x}_i$
fim se // sem melhorias da *fitness*
se $f(\boldsymbol{x}_i) < f(\boldsymbol{g}_{best})$ **então** // atualizar melhor indivíduo da vizinhança
$\boldsymbol{g}_{best} = \boldsymbol{p}_{best_i}$
fim se // sem melhorias do melhor global
para cada variável j da partícula i **faça** // limitar posição
se $x_{i,j} > \overline{x}$ **então**
$x_{i,j} = \overline{x}$
fim se
se $x_{i,j} < \underline{x}$ **então**
$x_{i,j} = \underline{x}$
fim se
fim para
fim para
enquanto condição de término não atingida

Uma breve descrição do funcionamento do algoritmo é a seguinte: Inicialmente uma partícula é identificada como a melhor da população, com base no valor calculado pela *fitness*. Todas as outras partículas são aceleradas na direção dessa partícula promissora, mas também em direção às suas próprias melhores posições previamente descobertas. Ocasionalmente, as partículas explorarão regiões além das partículas atualmente reconhecidas como melhores. Todas elas podem descobrir melhores posições durante a busca, caso em que as demais partículas são redirecionadas para essa

nova posição mais bem sucedida. Normalmente, uma boa solução está cercada por soluções igualmente boas, ou até melhores. Ao aproximar-se da melhor solução atual, a partir de diferentes direções no espaço de busca, aumentam as chances de alguma partícula localizar uma solução melhor, vizinha à antiga melhor.

3.5 Enxame de Partículas Padronizado

O PSO se tornou uma meta heurística comum na comunidade de otimização. Um padrão foi definido com objetivo de estabelecer uma extensão da versão original do algoritmo. Seu propósito final é servir de referência para comparação do desempenho de novas técnicas propostas. A padronização do algoritmo inclui:

- Topologia local em anel;
- Uso da equação com coeficiente de constrição;
- 50 partículas;
- Inicialização não uniforme da população e
- Não avaliação de partículas situadas para além dos limites do espaço de busca.

A principal vantagem do modelo *lbest* é sua taxa de convergência mais lenta com relação ao modelo *gbest* tradicional. Convergência muito rápida normalmente não é desejável pois pode prevenir o algoritmo de escapar de mínimos locais, ocasionando convergência prematura. A solução encontrada pelo modelo local frequentemente supera o valor obtido pelo modelo global, particularmente em problemas multimodais. A despeito das vantagens da topologia local, é importante notar que esse modelo não deve ser considerado como a escolha ótima em todos as situações. A taxa de convergência mais rápida da topologia global pode resultar em soluções melhores em problemas unimodais simples, devido à ausência dos perigos de convergência em mínimos locais subótimos. Em razão da velocidade de convergência lenta do modelo *lbest*, é requirido um número maior de avaliações da função objetivo.

Nessa abordagem é permitido que as partículas ultrapassem os limites do espaço, mantendo suas velocidades e posições inalteradas. Não é permitido que posições inviáveis sejam aceitas como melhor solução individual ou global. Segundo esse método, as partículas localizadas fora do espaço de busca irão, eventualmente, retornar

à região permitida em consequência da influência dos vizinhos. Muitos algoritmos de otimização são propensos à convergência em direção à região do espaço de busca na qual a população é inicializada, influenciando no desempenho do algoritmo. Com a finalidade de evitar esse viés, o procedimento de inicialização assimétrica é adotado.

3.6 *Fully Informed Particle Swarm*

O algoritmo *particle swarm* funciona por meio de influência entre as partículas de sua população. Em suas primeiras versões, era assumido que todas as partículas deveriam receber a melhor informação disponível e, além disso, cada partícula deveria ser influenciada pelo melhor ponto encontrado pelo melhor membro da sua vizinhança. Estudos recentes sugerem que pressupor essa ideia não é necessário. Esta seção apresenta uma alternativa às versões originais conceitualmente mais concisa ao mesmo tempo em que mantém, e às vezes melhora, o desempenho do algoritmo PSO tradicional. Nesta versão, nomeada *Fully Informed Particle Swarm*, abreviada para FIPS, as partículas utilizam informações provenientes de todos os seus vizinhos, e não apenas do melhor indivíduo presente na vizinhança.

Desenvolvido inicialmente por Mendes, Kennedy e Neves em 2004 e posteriormente aperfeiçoado por Kennedy e Mendes (2006), o FIPS é baseado na ideia de que cada partícula é influenciada por todos os indivíduos de sua vizinhança de uma maneira bem específica (JORDAN; HELWIG; WANKA, 2008). Todos os melhores pontos encontrados por cada membro da vizinhança são usados para atualizar a velocidade das partículas. As partículas possuem, consequentemente, conhecimento a respeito das melhores soluções encontradas até determinado momento por qualquer um de seus vizinhos, ao invés de acompanhar somente a partícula mais bem sucedida de sua vizinhança, detentora momentaneamente da melhor solução encontrada. Contrariamente ao estabelecido pelo algoritmo PSO canônico, não há, no FIPS, individualismo.

Essa abordagem é uma simplificação de todo o aspecto social e psicológico da sociedade humana em que indivíduos não são afetados exclusivamente por aqueles mais bem sucedidos, persuasivos ou prestigiosos. É mais eficiente pensar a vizinhança social como um modelo em que seus indivíduos são afetados pelo estado atual dos membros próximos de sua rede social ao invés de serem influenciados unicamente pelo desempenho do melhor indivíduo (GRANOVETTER, 1973; LATANE, 1981).

A diferença fundamental existente entre o PSO original e o FIPS refere-se ao fato

de que neste último os indivíduos congregam informações a respeito de toda a vizinhança; existe uma forte influência social no algoritmo FIPS. O modelo do FIPS é dado pela seguinte equação (MENDES; KENNEDY; NEVES, 2004), (KENNEDY; MENDES, 2006):

$$\boldsymbol{v}_i(t+1) = \lambda \cdot \left[\boldsymbol{v}_i(t) + \sum_{n=1}^{\mathcal{N}_i} \frac{U(0,\varphi) \cdot (\boldsymbol{p}_{nbr(n)} - \boldsymbol{x}_i(t))}{\mathcal{N}_i} \right] \tag{3.6}$$

em que $\mathcal{N}_i$ representa o número de vizinhos que a partícula i possui e $nbr(n)$ representa seus n vizinhos. As posições das partículas ainda são atualizadas de acordo com a equação (3.2). Na versão do algoritmo FIPS apresentada nesta seção, uma determinada partícula não influencia a si própria, somente é usada a discrepância entre sua posição corrente e as posições das melhores soluções de seus vizinhos. Considerou-se uma abordagem bem estabelecida na literatura, investigada por Kennedy e Mendes (2006), em que cada partícula possui exatamente três vizinhos. O método FIPS é mostrado no Algoritmo 2.

Algoritmo 2 FIPS

Parâmetros de entrada: Tamanho P da população.
// inicialização aleatória da população de partículas usando distribuição de probabilidade
// uniforme, em que $\underline{x}$ e $\overline{x}$ são os limites inferior e superior respectivamente.
para cada partícula i **faça**
  $\boldsymbol{x}_i = \underline{x} + (\overline{x} - \underline{x}) \cdot U(0,1)$
  $\boldsymbol{p}_{best_i} = \boldsymbol{x}_i$
fim para
$\boldsymbol{g}_{best} = \arg\min_{i=1,\ldots,p} \{f(\boldsymbol{x}_i)\}$ // melhor indivíduo da população
para cada partícula i **faça**
  carregar $nbr(i)$ // estrutura da vizinhança
fim para
repetir
  para cada partícula i **faça**
    para cada dimensão j da partícula i **faça** // atualizar posição $\boldsymbol{x}_i$ segundo (3.6)
      $x_{sum} = \sum_{n=1}^{\mathcal{N}_i} U(0,\varphi) \cdot (p_{nbr(n),j} - x_{i,j})$
      $v_{i,j} = \lambda \cdot \left(v_{i,j} + x_{sum}/\mathcal{N}_i\right)$
      $x_{i,j} = x_{i,j} + v_{i,j}$
    fim para
    se $f(\boldsymbol{x}_i) < f(\boldsymbol{p}_{best_i})$ **então** // atualizar melhor posição individual
      $\boldsymbol{p}_{best_i} = \boldsymbol{x}_i$
    fim se // sem melhorias da *fitness*
  fim para
enquanto condição de término não atingida

Segundo os estudos de Kennedy e Mendes (2006) o *fully informed* PSO é bastante

susceptível a alterações na topologia da população. Contudo, com uma boa escolha da topologia, esse método é capaz de superar o desempenho da versão canônica do seu predecessor (KENNEDY; MENDES, 2006). Mendes em 2004 realizou estudos sobre a metodologia *fully informed* e investigou pequenas modificações em relação ao método FIPS original. Mendes propôs que a contribuição de cada partícula da vizinhança fosse ponderada pela qualidade das soluções. Todavia, os resultados de simulação obtidos não confirmaram as expectativas. Os testes retrataram uma aparente inferioridade dessa abordagem quando comparada com a versão que desconsidera os pesos das soluções encontradas por cada partícula durante o cálculo de novos pontos no espaço de busca. Além disso, houve um significativo aumento do custo computacional em decorrência da manipulação de variáveis contendo os valores da função objetivo (*fitness*) envolvidos nos cálculos. Uma outra abordagem analisada por Mendes em seu trabalho pondera as contribuições das partículas pela distância em que se encontram uma das outras no espaço de busca. Essa alternativa, entretanto, também não demonstrou ser eficaz.

A versão tradicional do algoritmo FIPS é uma generalização da versão canônica. De fato, caso $\mathcal{N}_i$ seja definido de modo a conter apenas a própria partícula i e seus melhores vizinhos, a equação (3.6) é equivalente à equação (3.4) apresentada na página 30. A velocidade de cada partícula é atualizada segundo um cálculo estocástico da média ponderada da diferença entre a posição atual da partícula e a melhor solução previamente localizada por cada um de seus vizinhos. O algoritmo não utiliza informações sobre a relevância da qualidade de cada solução encontrada pelos seus vizinhos e tampouco faz uso da atual melhor solução encontrada pela própria partícula. As partículas simplesmente oscilam ao redor do centro de gravidade da região definida pelos melhores pontos estabelecidos por seus vizinhos.

No PSO convencional, em que as partículas escolhem, entre seus vizinhos, o melhor indivíduo como fonte de influência e ignoram os demais, o tamanho da vizinhança está estritamente relacionado apenas com a quantidade de partículas disponíveis para escolha da melhor. Quanto maior a vizinhança, mais "influente" será a partícula escolhida. Na vizinhança *fully informed*, contudo, todos os vizinhos são fontes de influência. Desse modo, o tamanho da vizinhança determina o quão diversa será essa influência. Mendes, Kennedy e Neves (2004) forneceram diretrizes para auxiliar a decisão sobre qual configuração de vizinhança adotar, restringindo as opções a apenas um subconjunto de topologias apropriadas.

3.7 *Bare Bones Particle Swarm*

Após mais de 15 anos de pesquisas, iniciadas com o trabalho de Kennedy e Eberhart, o algoritmo *Particle Swarm Optimization* (PSO) se consolidou como uma técnica de otimização baseada em inteligência coletiva muito eficaz, sendo aplicada a uma ampla variedade de aplicações. Diversas melhorias no algoritmo já foram propostas. Entender o funcionamento do algoritmo PSO e seu processo de convergência foi, e continua sendo, objetivo de estudo de muito trabalhos na literatura (KENNEDY, 1998; EBERHART; SHI; KENNEDY, 2001; CLERC; KENNEDY, 2002).

Alguns pesquisadores estudaram uma versão que elimina o termo da posição corrente, e simplesmente posicionam a partícula em uma nova posição segundo experiências anteriores. No ano de 2003, Kennedy propôs aquela que é considerada por muitos a versão mais simples do PSO, denominada *Bare Bones Particle Swarm*, e abreviada para BBPSO. Nessa versão simplificada, uma partícula não se move a partir de sua posição corrente em cada iteração do algoritmo. Em vez disso, a trajetória das partículas é descrita como uma órbita cíclica centrada randomicamente ao redor da média ponderada, em cada dimensão, dos melhores resultados individuais e da melhor solução coletiva (global) encontrada pela vizinhança. As novas posições são amostradas com base em uma distribuição Gaussiana. Essa estratégia é baseada na escolha do melhor indivíduo da população ($\boldsymbol{g}_{best}$ ou $\boldsymbol{l}_{best}$) e no uso da informação sobre a melhor posição encontrada por cada partícula ($\boldsymbol{p}_{best_i}$). A média dessa distribuição é dada pelo ponto médio entre a melhor solução global e a melhor local. O desvio padrão é calculado como a diferença absoluta entre o $\boldsymbol{g}_{best}$ e a melhor posição de cada partícula. O *Bare Bones* PSO de Kennedy não necessita de ajustes de parâmetros, é facilmente implementado, reduz os custos computacionais envolvidos e é bastante eficiente. Apesar do algoritmo BBPSO ser uma evolução do PSO canônico, apenas no trabalho (PAN et al., 2008) ficou comprovado que é possível deduzí-lo matematicamente a partir do PSO. Uma fórmula geral para o BBPSO foi apresentada por Pan et al..

Um dos mistérios da cognição humana é o processo pelo qual as pessoas são capazes de gerar novas ideias. Campbell (1960) indicou a possibilidade do ser humano adotar uma postura darwinista em um processo de variação cega e retenção seletiva na criação e aperfeiçoamento de novos pensamentos. Entretanto, Campbell não foi capaz de sugerir como seria o funcionamento da "variação cega" mencionada. Pode-se considerar que a criação de novas ideias seja regida por um processo aleatório

análogo à mutação – contudo não há evidências empíricas na psicologia humana que sustentem a hipótese de um "gerador randômico de ideias". A distribuição Gaussiana é frequentemente observada na natureza. Essa distribuição (ou qualquer outra aproximação em formato de sino) possui características positivas para solução de diversos problemas. Primeiro, o pico central do sino descreve seu aspecto conservador; a maioria dos pontos serão gerados na proximidade de valores anteriores. Segundo, sua cauda longa indica a habilidade para geração de *outliers*. Isso é necessário para descoberta de novas soluções. *Outliers* devem ser gerados ocasionalmente para manter vivo o espírito de exploração, sem, contudo, impedir a investigação de boas regiões.

Em seu trabalho (KENNEDY, 2003), Kennedy reportou um histograma da frequência dos pontos testados com o *particle swarm* canônico. A Figura 3.2 representa uma distribuição em formato clássico de sino centrado exatamente entre a melhor solução individual encontrada até o momento e a melhor solução global encontrada pela população – muito importante na exploração de novas regiões no espaço de busca. A extensão da curva em sino do gráfico aparenta relacionar-se com a distância entre os dois pontos.

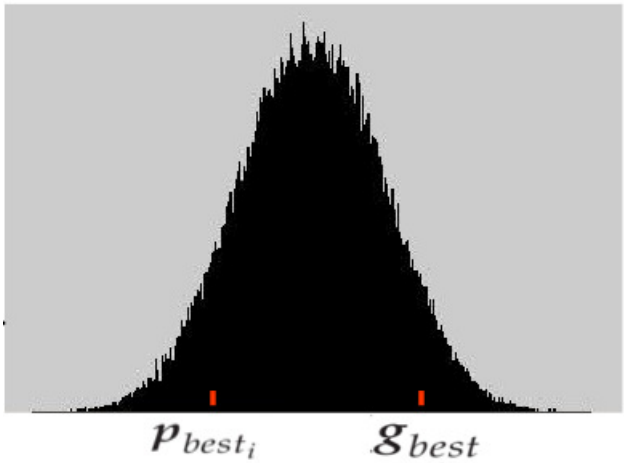


Figura 3.2: Histograma da frequência de pontos testados com PSO. Quando p_{best_i} e g_{best} são mantidos constantes, o PSO canônico amostra o espaço de busca segundo uma distribuição em formato de sino centrada exatamente entre p_{best_i} e g_{best}.

A natureza exata da distribuição não é clara: tem formato de sino assim como a distribuição Gaussiana mas, com extremidades mais prolongadas, assemelha-se à distribuição de Cauchy. Ao analisar um histograma semelhante da Figura 3.2, Kennedy (2003) sugeriu a remoção de toda a parcela da velocidade da equação do *particle swarm*. A principal fundamentação subjacente a essa abordagem, ao simplesmente substituir a equação de movimento das partículas por ruído Gaussiano, é a de que

o $\boldsymbol{p}_{best_i}$ é, por si só, uma boa solução; cada indivíduo mantém um pouco de sua "personalidade". O sucesso do algoritmo tem sido atribuído a duas características: a habilidade de "sobrevoar" boas soluções e, por conseguinte, explorar novas regiões promissoras; e o passo de tamanho adaptativo.

O BBPSO utiliza distribuição Gaussiana para amostrar o espaço de busca com base na melhor solução global ($\boldsymbol{g}_{best}$) e na melhor solução individual de cada partícula i ($\boldsymbol{p}_{best_i}$), de acordo com a seguinte equação:

$$x_{i,j} \sim N\left(\mu_{i,j}, \sigma_{i,j}^2\right) \tag{3.7}$$

em que $N(\mu_{i,j}, \sigma_{i,j}^2)$ representa a distribuição Gaussiana com média igual à $\mu_{i,j} = (gbest_j + pbest_{i,j})/2$ e desvio padrão $\sigma_{i,j} = \left|gbest_j - pbest_{i,j}\right|$. O algoritmo BBPSO é descrito com mais detalhes no Algoritmo 3. A equação (3.7) é usada em substituição às equações (3.2) e (3.4). Estudos teóricos (BERGH, 2002; BERGH; ENGELBRECHT, 2006) comprovaram a conjectura de que toda partícula da população converge para a média ponderada das melhores soluções individuais e coletiva.

Esse método possibilita que as partículas, cujas melhores soluções individuais encontradas até determinado momento estão afastadas da melhor solução global, dêem passos largos em direção à posição do $\boldsymbol{g}_{best}$. Isso permite que o $\boldsymbol{p}_{best_i}$ se aproxime do local onde se encontra o melhor mínimo da população. Quando as melhores soluções individuais se aproximam do $\boldsymbol{g}_{best}$, os passos ficam menores para limitar a diversificação em favor da intensificação. É importante ressaltar que, embora o BBPSO utilize informação global, esse algoritmo é uma versão amostrada do PSO, isto é, ao invés de as partículas se moverem segundo uma trajetória, elas amostram o espaço de busca de acordo com uma distribuição normal.

Poli e Langdon em 2007 aplicaram uma recém criada metodologia teórica com objetivo de melhor compreender o comportamento do algoritmo *Bare Bones* PSO. Poli e Langdon evitaram muitas das hipóteses pouco realistas frequentemente adotadas em análises de versões do PSO. Usando técnicas de elementos finitos, os autores configuraram um modelo discreto de Markov do BBPSO. A ideia era discretizar a função objetivo usando métodos de elementos finitos para produzir estados distintos correspondentes no algoritmo de busca. A discretização facilita a computação da matriz de transição para o sistema. Iterando as cadeias de transição da matriz, informações precisas são fornecidas a respeito do comportamento BBPSO em cada geração, incluindo a probabilidade de encontrar o ótimo global, o tempo esperado de execução

Algoritmo 3 BBPSO

Parâmetros de entrada: Tamanho P da população.
// inicialização aleatória da população de partículas usando distribuição de probabilidade
// uniforme, em que $\underline{x}$ e $\overline{x}$ são os limites inferior e superior respectivamente.
para cada partícula i **faça**
$\boldsymbol{x}_i = \underline{x} + (\overline{x} - \underline{x}) \cdot U(0,1)$
$\boldsymbol{p}_{best_i} = \boldsymbol{x}_i$
fim para
$\boldsymbol{g}_{best} = \arg\min_{i=1,\ldots,P} \{f(\boldsymbol{x}_i)\}$ // melhor indivíduo da população
repetir
para cada partícula i **faça**
para cada dimensão j da partícula i **faça** // atualizar posição $\boldsymbol{x}_i$ segundo (3.7)
$\mu = \left(gbest_j + pbest_{i,j}\right)/2$
$\sigma = \left|gbest_j - pbest_{i,j}\right|$
$x_{i,j} = \mu + \sigma \cdot N(0,1)$
fim para
se $f(\boldsymbol{x}_i) < f(\boldsymbol{p}_{best_i})$ **então** // atualizar melhor posição individual
$\boldsymbol{p}_{best_i} = \boldsymbol{x}_i$
fim se // sem melhorias da *fitness*
se $f(\boldsymbol{x}_i) < f(\boldsymbol{g}_{best})$ **então** // atualizar melhor indivíduo da vizinhança
$\boldsymbol{g}_{best} = \boldsymbol{p}_{best_i}$
fim se // sem melhorias da *fitness*
para cada variável j da partícula i **faça** // limitar posição
se $x_{i,j} > \overline{x}$ **então**
$x_{i,j} = \overline{x}$
fim se
se $x_{i,j} < \underline{x}$ **então**
$x_{i,j} = \underline{x}$
fim se
fim para
fim para
enquanto condição de término não atingida

entre outras informações. Além disso, é interessante também a investigação sobre como é afetado o desempenho do algoritmo ao adotar diferentes distribuições. Poli e Langdon consideram as seguintes: distribuição uniforme, distribuição Gaussiana e distribuição de Cauchy.

4 *Estratégia de Salto*

"Dream the dream of a distant place
sheltered and hidden from the human race
Long ago leaving everything behind
out in the coldness to find peace of mind

Leaving ground destination is unknown
into the darkness and far away from home
Will your dream come true and
what will you find when fate is your guide."

Riding On Fire
Iron Savior

Todas as versões do algoritmo PSO apresentadas anteriormente sofrem com a otimização de funções com muitos mínimos locais em espaços de busca de alta dimensão. Neste capítulo será apresentada uma técnica, denominada estratégia de salto (*jump*), cujo objetivo é evitar que o algoritmo convirja prematuramente em direção a mínimos locais.

Com o intuito de investigar o desempenho dessa abordagem, testes experimentais foram conduzidos usando os vários algoritmos populacionais descritos neste trabalho: o PSO canônico com topologia global, PSO padrão com topologia local em anel, FIPS e BBPSO. Os resultados sugerem que a estratégia de *jump* proporciona uma grande melhoria no desempenho de todos os algoritmos em todos os problemas adotados.

4.1 Motivação

O PSO é um algoritmo populacional muito poderoso e com grande potencial para solução de problemas de otimização. Entretanto, a maioria das versões desenvolvidas nos últimos anos apresenta dificuldades na otimização de funções com muitos mínimos locais em espaços de alta dimensão.

Baseado no fato de que as partículas não necessitam seguir sempre o mesmo regime, Krohling (2005) iniciou o desenvolvimento de uma estratégia simples para evitar a convergência prematura do algoritmo original. Essa abordagem foi denominada de estratégia de *jump*. Seu objetivo é permitir que as partículas da população "saltem" pelo espaço de busca, sempre que necessário, na tentativa de escapar de mínimos locais. Essa estratégia é usada somente quando há indícios de estagnação das partículas, ou seja, quando não ocorrem melhorias no valor da função objetivo.

Neste trabalho, a ideia apresentada por (KROHLING, 2005) é estudada e estendida. Alguns dos algoritmos populacionais mais conhecidos da literatura foram usados para análise do desempenho da abordagem investigada. Convencionou-se usar, inicialmente, as distribuições de probabilidades Gaussiana e de Cauchy para geração de novas posições no espaço de busca para as partículas estagnadas. Em geral, a distribuição de probabilidade uniforme é utilizada para gerar números randômicos em muitas versões do PSO. Entretanto, novas abordagens estão sendo elaboradas com a adoção de distribuições como a Gaussiana e a Cauchy para geração de números aleatórios usados na equação de atualização da velocidade das partículas do *particle swarm*. O trabalho (KROHLING; COELHO, 2006) é um bom exemplo disso. Resultados de estudos recentes indicam que tanto a primeira quanto a segunda distribuição são boas escolhas para melhorar o desempenho do algoritmo PSO padrão.

As influências do uso de diferentes distribuições de probabilidades para geração dos coeficientes cognitivos-sociais necessários ao PSO foram investigadas em (COELHO; KROHLING, 2005; KROHLING, 2004; KROHLING, 2005). O *particle swarm* padrão é essencialmente dinâmico e, portanto, não facilmente simulado com pontos amostrados de distribuições de probabilidades estáticas (KENNEDY, 2003; KENNEDY, 2004; KENNEDY, 2005a). As evidências sugerem que, como um ponto no espaço de busca depende dos pontos anteriores e da direção do movimento, as partículas movem-se de maneira oscilatória ao redor do centro de suas melhores posições anteriores.

4.2 Algoritmos Coletivos com Estratégia de Saltos usando Distribuições de Probabilidades

Nesta seção será apresentada a versão híbrida dos algoritmos investigados com a estratégia de *jump* utilizando as distribuições de probabilidade Gaussiana e de Cauchy. Foram escolhidas as seguintes versões do PSO: PSO com topologia global, PSO com topologia local em anel, FIPS, and BBPSO. Essas quatro versões são bastante representativas. Essa abordagem foi inicialmente proposta por Krohling e Mendel em 2009.

Na estratégia de *jump*, o movimento das partículas é governado por dois regimes dinâmicos. Enquanto houver melhorias na otimização da função objetivo, as partículas são movidas segundo as equações originais dos respectivos algoritmos. Contudo, ao menor sinal de estagnação do algoritmo, as partículas em estagnação passam a se movimentar (ou amostrar o espaço de busca) de acordo com um segundo regime dinâmico, o regime de *jump*. A introdução da estratégia de *jump* possibilita que novos pontos, em regiões promissoras, sejam amostrados. O objetivo é escapar de atratores locais. No contexto do PSO, o termo *jump* é mais apropriado que mutação porque as partículas se movimentam ("voam") pelo espaço de busca ao invés de evoluírem.

A cada etapa do processo de otimização, o valor da *fitness* de cada partícula é avaliado. Se não houver melhorias da função objetivo, um parâmetro para controle de estagnação da partícula é incrementado em uma unidade até que esse valor atinja um número máximo pré-especificado, denominado *maximum_stagnation_interval* – intervalo máximo de estagnação. Quando isso acontece é sinal de estagnação e, portanto, essa partícula deve "saltar" para uma nova posição em uma região mais promissora para continuar sua busca por melhores soluções. Esse salto é controlado pela introdução de um operador usando dois dos operadores mais comuns à comunidade de algoritmos evolutivos: mutação Gaussiana e de Cauchy (YAO; LIU, 1996; YAO; LIU; LIN, 2002; RUDOLPH, 2001). A nova posição da partícula é, então, dada por:

$$\boldsymbol{x}_i(t+1) = \boldsymbol{p}_{best_i} \cdot (1 + \eta N(0,1)) \tag{4.1}$$

ou

$$\boldsymbol{x}_i(t+1) = \boldsymbol{p}_{best_i} \cdot (1 + \eta C(0,1)) \tag{4.2}$$

em que η denota o parâmetro de escala. Na equação (4.1), $N(0,1)$ representa um número randômico gerado segundo a distribuição Gaussiana, dada por:

$$f_p(x;\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma}} \cdot e^{\frac{-(x-\mu)^2}{2\sigma^2}} \tag{4.3}$$

em que μ é a média e σ^2 é o desvio padrão.

Na equação (4.2), $C(0,1)$ também representa um número aleatório, porém gerado de acordo com a distribuição de probabilidade de Cauchy com parâmetro de escala t centrado na origem s, como descrito a seguir:

$$f_p(x;s,t) = \frac{1}{\pi t} \cdot \left[1 + \left(\frac{x-s}{t}\right)^2\right]^{-1} \tag{4.4}$$

Em ambos os casos, são gerados números aleatórios diferentes para cada variável $j = 1,\ldots,n$ da cada partícula i. Os algoritmos populacionais combinados com a estratégia de *jump* para escapar de mínimos locais usando *jumps* Gaussianos ou de Cauchy são apresentados no Algoritmo 4.

Como pode ser observado no Algoritmo 4, sempre que uma partícula i ultrapassa os limites (superior e inferior) do espaço de busca, ao invés de seguir o padrão estabelecido pelo PSO original, essa partícula é recolocada na mesma posição do seu $\boldsymbol{p}_{best_i}$. Notou-se experimentalmente que essa pequena alteração no código do algoritmo influencia positivamente a convergência de todas as versões do PSO investigadas com *jump*.

4.3 Experimentos Preliminares e Comparações

Esta seção descreve os experimentos realizados com os algoritmos populacionais discutidos no trabalho implementados com a estratégia de *jump*, usando diversas funções de testes (*benchmarks*) bem estabelecidas na literatura. Embora essas funções não forneçam, necessariamente, uma indicação precisa do desempenho de um algoritmo em problemas reais, elas são bastante utéis ao processo de investigação de alguns aspectos dos algoritmos considerados.

4.3.1 Problemas de Otimização Multimodais

Assumindo que o principal objetivo da introdução da estratégia de *jump* é escapar de mínimos locais para evitar que as partículas da população convirjam precoce-

Algoritmo 4 PSO usando estratégia de saltos

Parâmetros de entrada: Tamanho P da população, parâmetro de escala η e o intervalo máximo de estagnação (*maximum_stagnation_interval)*.
// inicialização aleatória da população de partículas usando distribuição de probabilidade
// uniforme, em que $\underline{x}$ e $\overline{x}$ são os limites inferior e superior respectivamente.
para cada partícula i **faça**
$\boldsymbol{x}_i = \underline{x} + (\overline{x} - \underline{x}) \cdot U(0,1)$
$\boldsymbol{p}_{best_i} = \boldsymbol{x}_i$
fim para
$g_{best} = \arg\min_{i=1,\ldots,P}\{f(\boldsymbol{x}_i)\}$ // melhor indivíduo da população
repetir
para cada partícula i **faça**
se $stagnation\_interval_i \leq maximum\_stagnation\_interval$ **então**
atualizar posição $\boldsymbol{x}_i$ de acordo com (3.2) e (3.4). // PSO
ou atualizar posição $\boldsymbol{x}_i$ de acordo com (3.6) e (3.2). // FIPS
ou atualizar posição $\boldsymbol{x}_i$ de acordo com (3.7). // BBPSO
else
atualizar posição $\boldsymbol{x}_i$ de acordo com (4.1) // distribuição Gaussiana
ou atualizar posição $\boldsymbol{x}_i$ de acordo com (4.2) // distribuição de Cauchy
$stagnation\_interval_i = 0$ // reiniciar intervalo de estagnação
fim se
se $f(\boldsymbol{x}_i) < f(\boldsymbol{p}_{best_i})$ **então** // atualizar melhor posição da partícula
$\boldsymbol{p}_{best_i} = \boldsymbol{x}_i$
else // incrementar o intervalo de estagnação em uma unidade
$stagnation\_interval_i += 1$
fim se // sem melhorias da *fitness*
// atualizar a melhor partícula da vizinhança (se necessário)
para cada variável j da partícula i **faça** // limitar posição
se $x_{i,j} > \overline{x}$ **então**
$x_{i,j} = pbest_{i,j}$
fim se
se $x_{i,j} < \underline{x}$ **então**
$x_{i,j} = pbest_{i,j}$
fim se
fim para
fim para
enquanto condição de término não atingida

mente, adotou-se um conjunto de funções *benchmarks* multimodais estabelecidas na literatura especializada (LEE; YAO, 2004; KROHLING; MENDEL, 2009). Essas funções, numeradas de g_1 até g_6, estão descritas na Tabela 4.1 e correspondem às funções f_4 até f_9, respectivamente apresentadas na Tabela 1 (LEE; YAO, 2004, p. 4).

Tabela 4.1: Funções com muitos mínimos locais, S determina os limites inferior e superior do espaço de busca, I representa o intervalo de inicialização assimétrico e g_{min} contém o valor do mínimo global. A dimensão de todos os problemas é dada por $n = 30$.

Funções de Teste	S	I	g_{min}
$g_1(\boldsymbol{x}) = -\sum_{i=1}^{n} x_i \sin(\sqrt{x_i})$	$(-500,500)^n$	$(-500,-250)^n$	-12569.5
$g_2(\boldsymbol{x}) = \sum_{i=1}^{n} \left\{ x_i^2 - 10\cos(2\pi x_i) + 10 \right\}$	$(-5.12,5.12)^n$	$(2.56,5.12)^n$	0.0
$g_3(\boldsymbol{x}) = -20\exp\left\{ -0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2} \right\} - \exp\left\{ \frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i) \right\} + 20 + e$	$(-32,32)^n$	$(16,32)^n$	0.0
$g_4(\boldsymbol{x}) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$(-600,600)^n$	$(300,600)^n$	0.0
$g_5(\boldsymbol{x}) = \frac{\pi}{n}\left\{ 10\sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i-1)^2\left\{1+10\sin^2(\pi y_{i+1})\right\}\right\} + (y_n-1)^2\Big\} + \sum_{i=1}^{n} u(x_i,10,100,4)$ $y_i = 1 + (1/4)\cdot(x_i+1)$ $u(x,a,k,m) = \begin{cases} k\cdot(x-a)^m, & x > a \\ 0, & -a \le x \le a \\ k\cdot(-x-a)^m, & x < -a \end{cases}$	$(-50,50)^n$	$(25,50)^n$	0.0
$g_6(\boldsymbol{x}) = 0.1\Big\{ \sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i-1)^2\left\{1+\sin^2(3\pi x_{i+1})\right\} + (x_n-1)^2\left\{1+\sin^2(2\pi x_n)\right\}\Big\} + \sum_{i=1}^{n} u(x_i,5,100,4)$	$(-50,50)^n$	$(25,50)^n$	0.0

4.3.2 Configuração Experimental

Vários algoritmos populacionais de otimização, incluindo o PSO, tendem a apresentar melhor desempenho quando o ótimo da função está localizado no, ou próximo do, centro de inicialização das partículas (normalmente a origem do espaço). Em muitos casos essa característica pode representar um problema em potencial durante o desenvolvimento e validação de uma metodologia nova; em várias funções *bechmarks* o ótimo global está contido nas proximidades da origem. Para dificultar o processo de otimização e reduzir uma possível tendência das partículas de convergirem para a região no espaço onde elas foram inicializadas, convencionou-se utilizar intervalos assimétricos não uniformes, apresentados na Tabela 4.1, para inicialização das partículas da população. Para cada função, o intervalo foi escolhido de maneira a não conter a região do mínimo global. Essa metodologia, por forçar que a população

expanda a busca para além de seus limites iniciais, é considerada um padrão em pesquisas e testes de desempenho e comparação de algoritmos (BRATTON; KENNEDY, 2007). Foi inicialmente proposta por (GEHLHAAR; FOGEL, 1996) e popularizada posteriormente por (ANGELINE, 1998).

É de conhecimento da comunidade de *particle swarm* (BRATTON; KENNEDY, 2007) que o PSO obtém bons resultados quando utiliza populações com número de indivíduos entre 20 e 100. Em todos os experimentos apresentados nesta seção, foram utilizadas 50 partículas, conforme sugerido por (LEE; YAO, 2004). Embora o tamanho ótimo da população seja uma variável dependente de cada problema, está fora do escopo deste trabalho investigar e analisar qual o melhor número de partículas para cada caso. Os experimentos foram executados 50 vezes e cada execução é interrompida somente quando um número máximo de iterações é alcançado (1500). Nos casos em que os valores de simulação foram inferiores à 10^{-8}, eles foram arredondados para 0.0.

Os valores do parâmetro η (fator de escala) para cada função foram obtidos a partir de simulações iniciais e são apresentados na Tabela 4.2. Um regra simples para encontrar bons valores iniciais para η é discutida em (KROHLING, 2005).

Tabela 4.2: Valores do parâmetro η usados nos experimentos.

Função	Parâmetro de escala η
g_1	20
g_2	1.1
g_3	1.1
g_4	1.1
g_5	1.1
g_6	0.1

O parâmetro *maximum_stagnation_interval*, usado para monitorar o valor da função objetivo (ou seja, se há ou não melhoria na *fitness*), foi fixado em cinco. Todavia, outros valores podem ser usados. Notou-se experimentalmente que o algoritmo híbrido não é muito sensível a pequenas variações no número de iterações consecutivas de estagnação das partículas antes de alternar do regime padrão para o esquema de *jump*. Outros critérios para monitoramento da *fitness* podem ser adotados.

Tabela 4.3: Resultados experimentais usando BBPSO, BBPSO+G_dJ (distribuição de probabilidade Gaussiana), BBPSO+C_dJ (distribuição de probabilidade de Cauchy). Média de 50 execuções.

Função	Método	Média	Desvio Padrão	Mediana	Melhor	Pior	Saltos Bem Sucedidos (%)
g_1	BBPSO	-10094.1	272.3	-10082.3	-10674.5	-9371.65	n/a
	BBPSO+G_dJ	-12472.2	153.021	-12569.5	-12569.5	-11856.4	3.16
	BBPSO+C_dJ	-12426.7	136.627	-12451	-12569.5	-11977.3	2.06
g_2	BBPSO	56.6927	17.9018	53.7277	33.8286	126359	n/a
	BBPSO+G_dJ	1.1689	4.006	0.0	0.0	23.879	1.36
	BBPSO+C_dJ	0.0	0.0	0.0	0.0	0.0	4.89
g_3	BBPSO	1.1865	4.5167	0.0	0.0	19.5566	n/a
	BBPSO+G_dJ	0.0	0.0	0.0	0.0	0.0	5.33
	BBPSO+C_dJ	0.0	0.0	0.0	0.0	0.0	17.27
g_4	BBPSO	0.0109783	0.0146363	0.00985728	0.0	0.0882895	n/a
	BBPSO+G_dJ	0.0	0.00634	0.0	0.0	0.0369	2.39
	BBPSO+C_dJ	0.0	0.0	0.0	0.0	0.0	8.71
g_5	BBPSO	0.0767712	0.172767	0.0	0.0	0.936053	n/a
	BBPSO+G_dJ	0.0352	0.0614	0.0	0.0	0.310	0.18
	BBPSO+C_dJ	0.0103	0.0314	0.0	0.0	0.103	0.69
g_6	BBPSO	0.00439495	0.00543734	0.0	0.0	0.0109874	n/a
	BBPSO+G_dJ	0.00351	0.00750	0.0	0.0	0.0439	7.36
	BBPSO+C_dJ	0.00395	0.00935	0.0	0.0	0.0439	9.72

4.3.3 Resultados e Discussões

Com o objetivo de investigar a eficácia da estratégia de *jump* incorporada à versão do BBPSO, comparou-se as médias dos resultados obtidos em todas as execuções do algoritmo original e de sua versão modificada. Todas as funções da Tabela 4.1 foram usadas nas comparações.

A Tabela 4.3 contém os resultados obtidos pelo BBPSO original e por sua versão com *jump*. Além de índices estatísticos como média dos melhores resultados coletados, desvio padrão, mediana, melhor e pior soluções encontradas, a Tabela 4.3 também apresenta o percentual (%) de *jumps* bem sucedidos, calculado dividindo-se o número de *jumps* que conduzem o algoritmo a regiões melhores pela quantidade total de *jumps* executados.

Observa-se pela Tabela 4.3 que BBPSO+C_dJ supera, em termos de métricas estatísticas, o algoritmo BBPSO+G_dJ nas funções g_2 e g_5. O desempenho do BBPSO com probabilidade Gaussiana é superior ao do BBPSO com distribuição de Cauchy somente na função g_1. Nas demais funções, ambos os métodos BBPSO+C_dJ e BBPSO+G_dJ

obtêm resultados similares, sem diferenças estatísticas significantes.

Como tentativa de melhorar ainda mais o desempenho da estratégia de *jump* apresentada, uma pequena alteração na abordagem foi testada: o algoritmo BBPSO com *jump* Gaussiano (e de Cauchy) foi implementado considerando o fator de escala η crescente. Contudo, os resultados obtidos com essa adaptação foram muito semelhantes aos obtidos pela estratégia com fator de escala fixo, indicando que aumentar o η periodicamente não proporciona melhores "saltos" (apesar de maiores).

As Figuras 4.1 até 4.6 ilustram os gráficos com a convergência dos algoritmos estudados para todas as funções testadas. Todos os métodos analisados apresentam bons resultados em termos de velocidade de convergência, apesar de não convergirem em todos os casos.

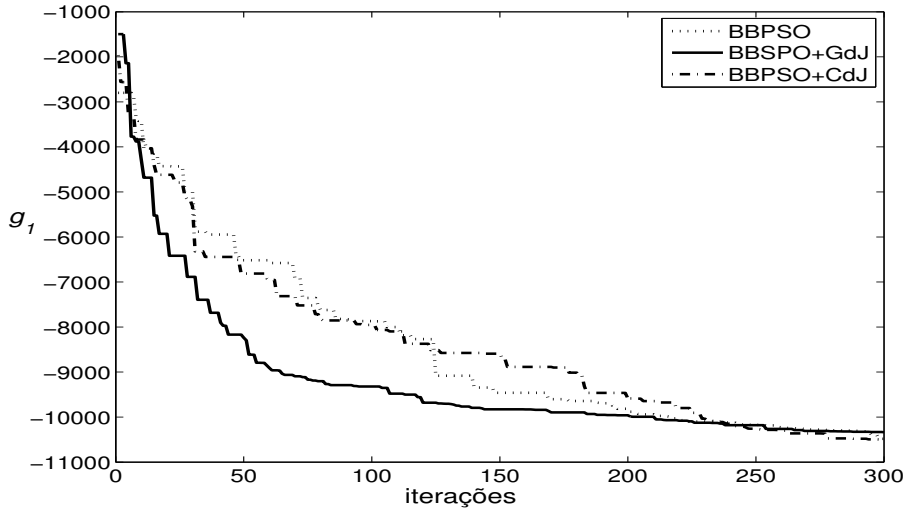


Figura 4.1: Convergência para função g_1

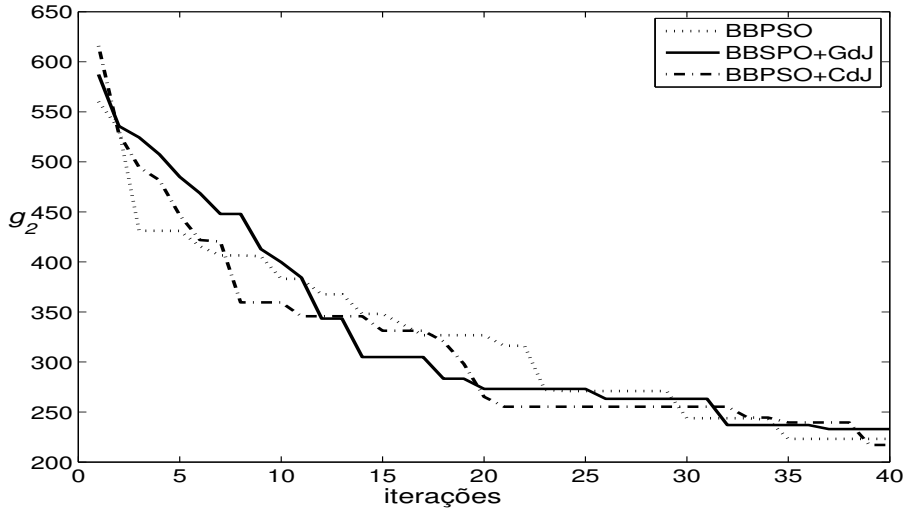


Figura 4.2: Convergência para função g_2

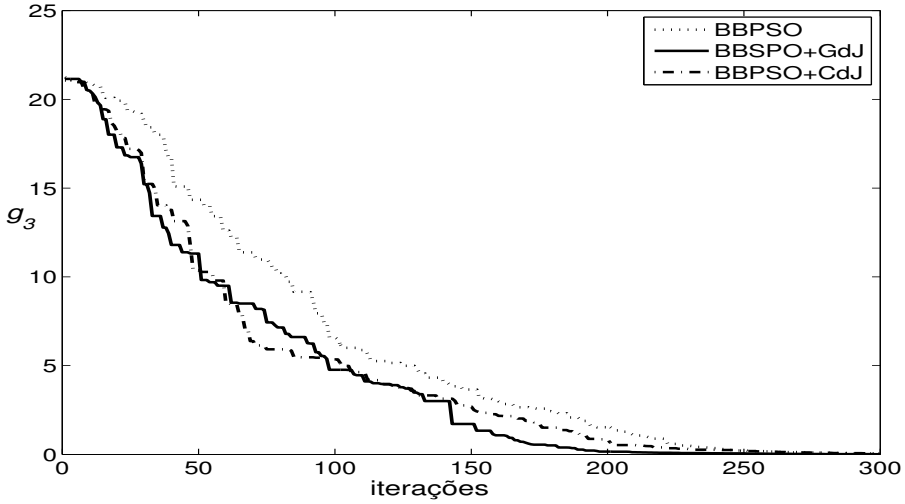


Figura 4.3: Convergência para função g_3

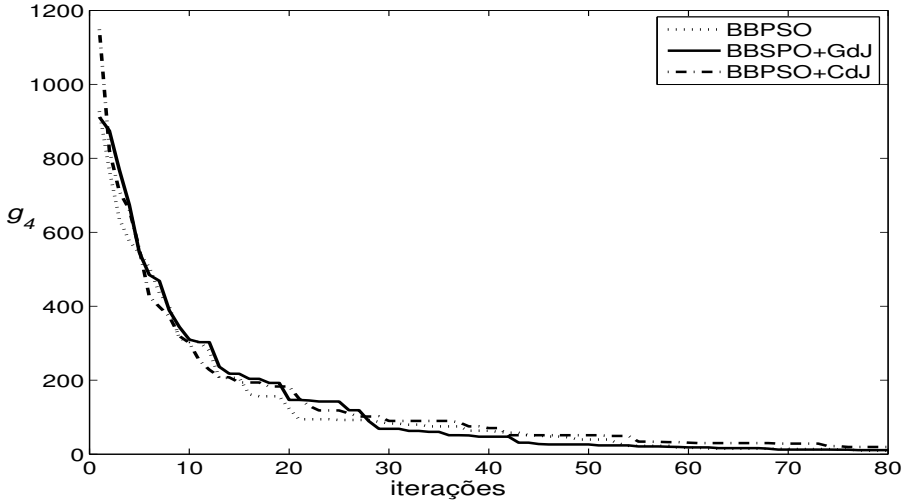


Figura 4.4: Convergência para função g_4

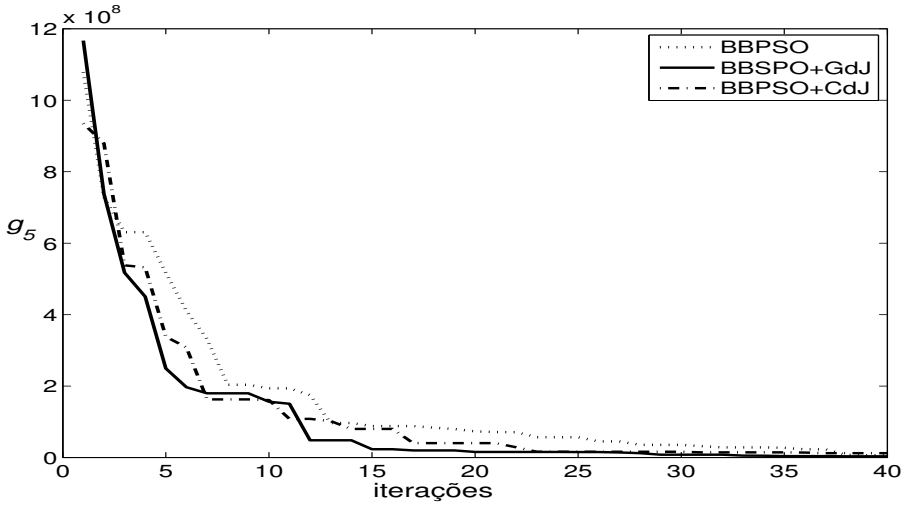


Figura 4.5: Convergência para função g_5

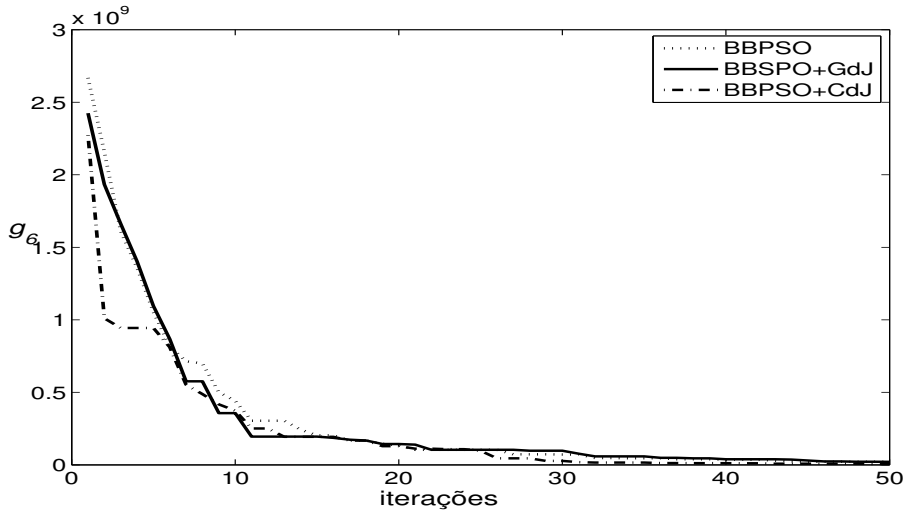


Figura 4.6: Convergência para função g_6

4.4 Conclusão

A introdução da estratégia de *jump* aos algoritmos populacionais melhora o desempenho dos algoritmos, uma vez que proporciona maiores oportunidades para as partículas escaparem de mínimos locais (KROHLING; MENDEL, 2009).

Essa abordagem é mais eficaz que todas as versões padrões do PSO discutidas anteriormente, com quase nenhuma adição de custo computacional. Além disso, essa estratégia pode ser facilmente implementada e aplicada a problemas do mundo real.

Os algoritmos populacionais combinados com a estratégia de *jump* apresentaram melhores valores médios em comparação com as versões originais (sem *jump*) em todas as funções adotadas nos experimentos. Além da superioridade apresentada pelos algoritmos modificados na média das 50 execuções, a abordagem híbrida também fornece valores de desvio padrão inferiores.

Com a realização dos experimentos e com a análise dos resultados, nota-se que um número significativo de *jumps* bem sucedidos acontece durante o estágio inicial do processo de otimização; o que é desejável visto que isso proporciona mais exploração. Após a fase de exploração do algoritmo, os *jumps*, em geral, não são tão eficazes quanto antes pois a tendência é que as partículas já estejam em uma região promissora.

5 *Caos e Algoritmos Evolutivos*

"When you know that your time is close at hand.
Maybe then you'll begin to understand,
Life down here is just a strange illusion."

Hallowed Be Thy Name
Iron Maiden

Neste capítulo será investigado um método híbrido baseado na estratégia de *jump* e no uso de mapas caóticos. Recentemente, diversas abordagens combinando algoritmos populacionais e evolutivos com mapas caóticos foram propostas. Alguns dos trabalhos mais relevantes no que tange ao uso de sequências caóticas com o intuito de impulsionar o desempenho desses algoritmos serão apresentados. Aspectos específicos sobre algumas das abordagens discutidas também serão analisadas de modo a destacar pequenas melhorias adotadas na estratégia estudada neste trabalho. Serão descritas de maneira bastante sucinta três das muitas sequências caóticas frequentemente aplicadas no contexto de otimização, especialmente em *particle swarm optimization*. Diversas propriedades estatísticas dos mapas os tornam atraentes para solucionar problemas de otimização.

O algoritmo híbrido estudado foi testado em um conjunto de funções multimodais bem conhecidas e estabelecidas na literatura. Seguindo a estratégia de *jump*, a versão caótica do algoritmo também trabalha em dois regimes. No primeiro, caso as partículas da população proporcionem melhorias na otimização da função objetivo, cada uma dessas partículas terá sua posição no espaço de busca atualizada de acordo com as equações convencionais do algoritmo PSO. No segundo regime, se não houver melhoria da *fitness*, então as partículas irão se mover de acordo com a estratégia de *jump*, porém, implementada usando mapas caóticos. Uma sequência caótica é usada na geração dos números randômicos necessários à equação da estratégia de *jump* (KROHLING; MENDEL; CAMPOS, 2011).

Os resultados de simulação indicam que a estratégia de *jump* impulsionam o desempenho dos algoritmos populacionais quando aplicados em problemas de otimização primordialmente multimodais.

5.1 Motivação

Neste capítulo são investigadas as características da dinâmica de caos no campo da inteligência coletiva (SI, do inglês *Swarm Intelligence*), um paradigma de inteligência distribuída. Esse paradigma possibilita que o comportamento coletivo de indivíduos simples, que interagem localmente entre si, proporcione a descoberta de informações globais consistentes. A inteligência surge do equilíbrio caótico entre individualidade e coletividade. O equilíbrio caótico é uma característica intrínseca de sistemas complexos.

O caos é um fenômeno que aparece com bastante frequência em sistemas não lineares. As suas principais características são: ergodicidade, estocasticidade e regularidade. Um sistema ergódigo é aquele no qual se pode prever eventos futuros por meio da estimação estatística de eventos passados. Quanto maior a ergodicidade, menor a incerteza apresentada pelo sistema. Essa propriedade pode ser uma alternativa interessante para promover diversidade em algoritmos estocásticos baseados em população. A teoria de caos é subjacente a muitos fenômenos naturais, de fluido de fluxo turbulento a padrões climáticos globais (GLEICK; GLAZIER; GUNARATNE, 1988), de ritmos cardíacos saudáveis (GOLDBERGER; RIGNEY; WEST, 1990) a sequências de códigos de DNA (OHNO, 1988). É natural supor que processos evolutivos possam ser compreendidos em termos da teoria de caos. O livro *Beyond Natural Selection* (WESSON, 1993) motiva a discussão à respeito dessa conjectura e a analisa cuidadosamente.

A grande parcela dos estudos realizados em inteligência coletiva propuseram várias melhorias baseadas no comportamento da busca caótica, comprovando a eficiência das sequências caóticas. Uma vasta quantidade de sequências distintas podem ser geradas simplesmente alterando-se as condições iniciais dos mapas. Ademais, essas sequências são determinísticas, reprodutíveis e sua implementação é bastante fácil. Recentemente, experimentos têm indicado que caos possui uma habilidade especial para evitar convergências prematuras. Por consequência, os mapas caóticos estão sendo intensamente adotados na área de otimização computacional, despertando o interesse da comunidade de inteligência evolutiva por algoritmos baseados em busca caótica.

5.2 Trabalhos Relacionados

Na área de otimização, o comportamento das sequências caóticas já foi explorado com algumas meta-heurísticas em problemas de otimização global com muitos mínimos locais. Esses métodos recorrem ao comportamento caótico na tentativa de localizar uma solução satisfatória em tempo hábil evitando sua convergência prematura devido à presença de mínimos indesejáveis. Caponetto et al. (2003) realizaram uma análise experimental sobre a convergência dos algoritmos evolutivos visando melhorar a eficácia desses algoritmos usando mapas caóticos. Essa abordagem é baseada na introdução de sequências caóticas, em vez de sequências aleatórias, durante todas as fases do processo evolutivo, isto é, desde a geração da população inicial à execução dos operadores de seleção, recombinação e mutação. Os experimentos sugerem que o uso de sequências caóticas pode proporcionar melhorias ao desempenho dos algoritmos, obtendo resultados superiores aos obtidos quando geradores de números randômicos são usados.

Coelho e Mariani (2008) propuseram uma combinação interessante entre o algoritmo inspirado nas colônias de formigas (ACA, do inglês *Ant Colony Algorithm*) e as sequências caóticas. Os autores argumentaram que a aplicação de sequências caóticas em substituição às sequências randômicas é uma estratégia eficaz na prevenção contra a convergência prematura do ACA pois diversifica a população do algoritmo.

Especificamente sobre *particle swarm* a maioria das abordagens propostas combinando PSO com mapas caóticos ou adicionam uma simples pertubação implementada como um operador de mutação aplicado à equação de atualização da velocidade, ou apenas substituem os números randômicos por sequências caóticas. Chuanwen e Bompard (2005), por exemplo, apresentaram um método híbrido do PSO baseado na introdução de caos na equação original do algoritmo. Há uma crença geral na comunidade de que os parâmetros φ_1 e φ_2 são um dos fatores principais para garantir uma boa convergência do PSO. Entretanto, visto que esses parâmetros não asseguram ergodicidade durante todo o processo de otimização pois são absolutamente aleatórios no algoritmo tradicional, Chuanwen e Bompard iniciaram uma investigação à respeito dos possíveis benefícios que o uso de mapas caóticos pode trazer ao PSO. O método foi testado e examinado em um sistema hidroelétrico. Os autores argumentaram que esse método é capaz de melhorar tanto a convergência quanto a precisão dos resultados.

O trabalho de (COELHO; HERRERA, 2007) apresentou um PSO caótico baseado no mapa Zaslavskii combinado com técnicas de agrupamento para identificação do modelo fuzzy de Takagi-Sugeno aplicado a um sistema não linear de movimento de ioiô. No contexto do trabalho, em virtude da ergodicidade e de propriedades dinâmicas das variáveis do mapa de Zaslavskii, o PSO utilizando busca caótica pôde escapar mais facilmente de mínimos locais do que o método tradicional.

Alatas, Akin e Ozer (2009) usaram um gerador de números caóticos sempre que um número aleatório era necessário ao algoritmo PSO clássico. Um total de oito mapas caóticos foram analisados e foi verificado que, em alguns casos, esses mapas melhoram a capacidade de busca global ao escapar de soluções locais.

Yang, Li e Cheng em 2007 consideraram as principais características estatísticas das sequências caóticas dos mapas logístico e Kent, e propuseram algoritmos híbridos combinando PSO e essas sequências caóticas. Após a realização de diversos experimentos com o algoritmo desenvolvido usado para solucionar problemas de otimização global, concluiu-se que a eficiência do método é diretamente influenciada pelas propriedades estatísticas apresentadas pelas sequências caóticas/estocásticas geradas pelos algoritmos caóticos/estocásticos.

Xiang, Liao e Wong (2007) apresentaram uma versão melhorada do algoritmo PSO combinada com o mapa caótico linear em trechos (*piecewise linear*). Eles analisaram as vantagens e as deficiências dessa metodologia. Resultados promissores foram obtidos. Vários aspectos incluindo a escolha do mapa, o método para mapeamento das variáveis e também a definição do intervalo da busca caótica foram avaliados no estudo. Para os autores, a busca caótica deveria apenas ser aplicada à melhor partícula da população pois a região em seu entorno pode ser a mais promissora. Essa região é estreitada à medida que o processo de otimização avança. Ou seja, o espaço de busca é reduzido enquanto a busca continua. Os autores argumentaram que a otimização caótica é mais eficiente em áreas menores.

Em (MENG et al., 2005), um mecanismo de busca caótica foi incorporado ao algoritmo *particle swarm* padrão de maneira adaptativa para evitar a estagnação das partículas da população, ao mesmo tempo que aumenta a velocidade de convergência do algoritmo. Esse método híbrido faz uso da ergodicidade presente na busca caótica para aumentar sua precisão e manter um equilíbrio entre a busca global e a local. Comparando-o com outros métodos, como o genético, os resultados de simulação do método proposto por Meng et al. sugerem uma superioridade significativa

com relação tanto à convergência quanto à robustez. Para possibilitar que as partículas escapem de mínimos locais, os autores propuseram substituir as partículas estagnadas (ou inativas) por novas criadas pela pesquisa caótica. Além disso, eles argumentaram que a busca caótica deve restringir-se a uma área pequena para almejar bons resultados com a pesquisa local. Cada raio de busca $R_{i,j}$ é definido com base no intervalo de inicialização de cada partícula $x_{i,j}$. O valor obtido pela busca caótica é, então, mapeado para $\left[x_{i,j} - R_{i,j}, x_{i,j} + R_{i,j}\right]$. A variável caótica $z_{n,d}$ é transferida para a variável de otimização $z'_{n,d}$, normalizando $z_{n,d}$ dentro da região centrada na posição atual da partícula com raio igual a $R_{i,j}$, de acordo com $z'_{n,j} = x_{i,j} + R_{i,j}\left(2z_{n,j} - 1\right)$. Embora os autores Meng et al. afirmem que seu algoritmo híbrido é capaz de resolver problemas complexos de otimização na medida em que evita a estagnação das partículas, nota-se experimentalmente que resultados melhores são obtidos quando o intervalo para geração das novas partículas é definido ao redor da melhor posição obtida até o momento pela partícula. O trabalho (MENG et al., 2005) não apresenta uma análise meticulosa sobre a influência do tamanho do raio de busca e como proceder para especificar um bom valor para esse raio.

Liu et al. (2005) desenvolveram um algoritmo *particle swarm* caótico combinando as habilidades de busca de estratégias evolutivas baseadas em população com o comportamento de pesquisas caóticas. A abordagem, denominada *Chaotic PSO*, consiste em uma estratégia iterativa de duas fases. As características do PSO são usadas em primeira instância para garantir diversificação durante a atualização das partículas da população, ao passo que os aspectos da dinâmica de caos são aplicados posteriormente para propiciar intensificação na medida em que modifica localmente a melhor partícula encontrada até o momento pelo PSO. Para manter a diversidade da população, Liu et al. propuseram que sejam geradas novas partículas randomicamente. Aconselha-se que a área para geração dessas novas partículas seja decrementada dinamicamente para assegurar uma taxa de convergência alta. Contudo, testes experimentais conduzidos no presente trabalho indicam que a redução gradual do espaço de busca não beneficia o algoritmo. O principal motivo reside no fato de que essa redução pode possibilitar a exclusão do mínimo global da área de busca resultante prematuramente. A cada iteração, o algoritmo desenvolvido em (LIU et al., 2005) constrói uma nova população formada por 4N/5 de novas partículas geradas aleatoriamente dentro do espaço de busca decrementado e N/5 de partículas existentes com substituição da melhor posição encontrada pelo resultado da busca caótica local. Resultados de simulações e comparações com outras meta-heurísticas indicaram que

essa abordagem pode aumentar a eficácia da busca e o desempenho final significativamente, embora admita algumas pequenas alterações. Seu custo computacional é elevado.

Aproveitando-se das vantagens apresentadas pelas variáveis caóticas, o trabalho de (YANG; NOMURA, 2008) introduziu a busca caótica no *quantum-behaved particle swarm optimization* (QPSO) com o objetivo de induzir diversidade na população durante a última fase da busca. Apesar do algoritmo QPSO ser uma alternativa viável para problemas de otimização, as partículas da população tendem a convergir umas às outras rapidamente; a perda de diversidade é inevitável. Entretanto, os resultados experimentais reportados pelos autores indicam que, com a inclusão da busca caótica, o QPSO é capaz de superar o desempenho do QPSO original e também do PSO tradicional. A busca caótica é realizada de acordo com a seguinte equação: $x_{i,j}(t+1) = x_{i,j}(t) + x_{max}(2z_d - 1)$. Yang e Nomura sugeriram que quando o valor de $x_{i,j}$ ultrapassar o intervalo $[-x_{max}, x_{max}]$, deve-se atribuir uma valor aleatório, dentro do intervalo do espaço de busca, para $x_{i,j}$ ao invés de fixar a posição dentro dos limites do intervalo de busca como no PSO padrão. Todavia, nos testes realizados no corrente trabalho, constatou-se que resultados melhores são obtidos quando a partícula i, ao ultrapassar os limites superior e inferior do espaço de busca, é recolocada na posição do seu $\boldsymbol{p}_{best_i}$.

Neste trabalho, é considerada uma abordagem com dois regimes dinâmicos. No primeiro regime, quando há melhoria da *fitness*, as partículas da população movem-se pelo espaço de busca segundo a estratégia tradicional do PSO, FIPS ou BBPSO. No segundo regime, porém, não havendo melhorias da função objetivo, as partículas consideradas estagnadas passam a se movimentar de acordo com a estratégia de *jump* implementada com base em mapas caóticos. Não obstante a existência de diferentes mecanismos de busca caótica na literatura especializada sobre algoritmos sócio-bio-inspirados, a ideia de uma estratégia híbrida de *jumps* caóticos e determinísticos pode ser considerada original. É importante notar que a sequência caótica não é usada dentro do PSO. Ao invés disso, adota-se a sequência caótica somente quando nenhuma melhoria é detectada.

5.3 Mapas Caóticos

Esta seção apresenta uma breve descrição das sequências caóticas usadas no trabalho. Matematicamente, sistemas não-lineares apresentam um comportamento determinístico não periódico e são muito sensíveis às condições iniciais, isto é, pequenas pertubações nas condições iniciais do sistema dinâmico provocam enormes variações. Os mapas caóticos são mapas que exibem comportamento caótico. Eles podem ser contínuos ou discretos e frequentemente surgem no estudo sobre sistemas dinâmicos. Os mapas discretos em geral possuem a forma de funções iterativas.

A seguir, são descritas algumas sequências caóticas frequentemente aplicadas no contexto de otimização, especialmente, *particle swarm optimization*. Investigou-se, resumidamente, os mapas: logístico, Gauss e Zaslawskii. Para maiores informações, sugere-se a consulta ao trabalho (LASOTA; MACKEY, 1994).

5.3.1 Mapa Logístico

O mapa logístico é descrito pela seguinte equação:

$$z_{k+1} = a z_k (1 - z_k) \tag{5.1}$$

em que $z_k \in (0;1)$. A Figura 5.1 ilustra um histograma para a sequência caótica gerada usando o mapa logístico. Diferentes valores iniciais para $z(0)$ são permitidos para geração da variável caótica z_k, com exceção dos seguintes pontos: 0.25, 0.50, 0.75 e 1.0.

5.3.2 Mapa Gaussiano

O mapa de Gauss é descrito por:

$$z_{k+1} = \frac{1}{z_k} - \left\lfloor \frac{1}{z_k} \right\rfloor \tag{5.2}$$

em que $z_k \in (0;1)$. A Figura 5.2 mostra um histograma para a sequência caótica gerada usando o mapa de Gauss, com estado inicial $z_0 = z(0)$ aleatório.

5.3.3 Mapa Zaslawskii

O mapa Zaslawskii é descrito por:

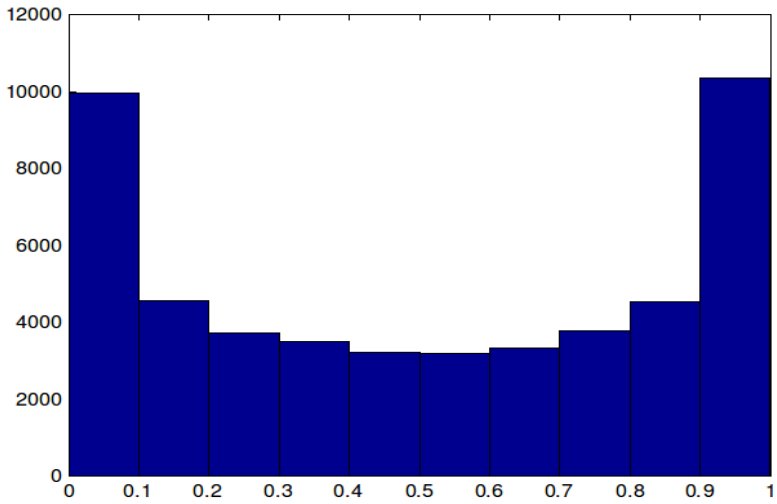


Figura 5.1: Histograma da sequência caótica gerada usando mapa logístico com $a = 4$ e estado inicial $z_0 = z(0)$ gerado randomicamente.

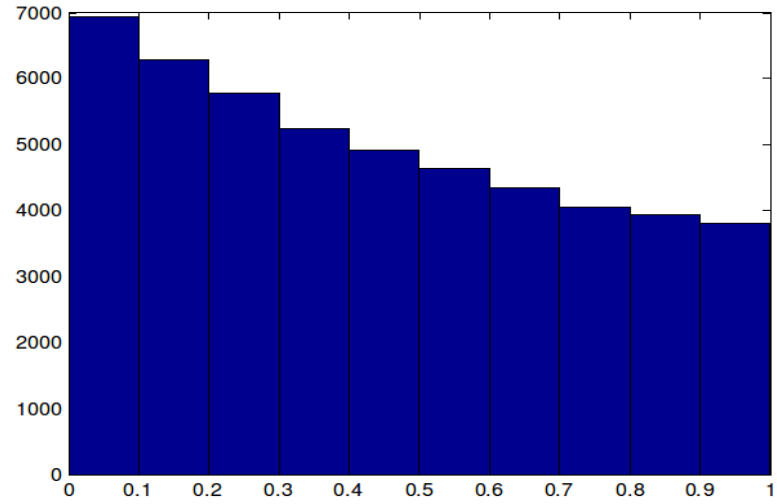


Figura 5.2: Histograma da sequência caótica gerada usando mapa gaussiano com estado inicial $z_0 = z(0)$ gerado randomicamente.

$$y_{k+1} = \cos(2\pi x_k) + e^{-r} y_k \tag{5.3}$$

$$z_{k+1} = (z_k + v + a y_{k+1}) \, mod(1) \tag{5.4}$$

A Figura 5.3 mostra um histograma para a sequência caótica gerada pelo mapa Zaslawskii.

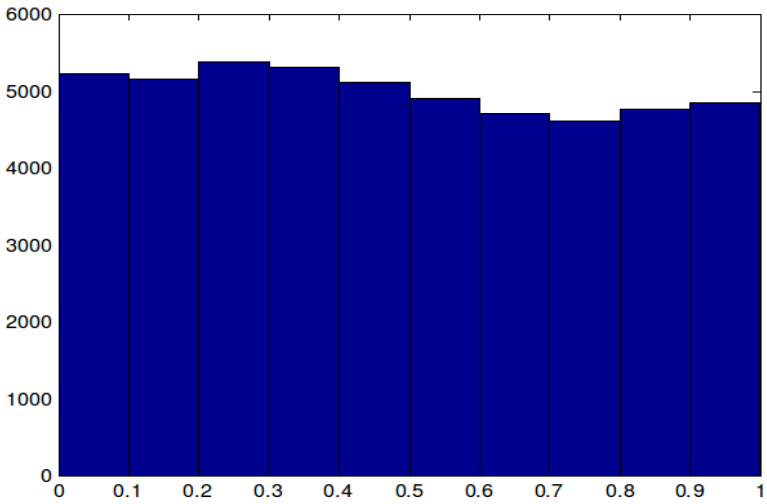


Figura 5.3: Histograma da sequência caótica gerada usando mapa Zaslawskii com $v = 400$, $a = 12$, $r = 3$ e estado inicial $z_0 = z(0)$ gerado randomicamente.

5.4 Enxame de Partículas com Saltos Caóticos

Nesta seção será apresentada uma versão híbrida dos algoritmos deste trabalho, semelhante à explicada no Capítulo 4. A diferença reside no fato de serem utilizados mapas caóticos para geração dos números randômicos necessários à equação da estratégia de *jump*. Conforme mencionado anteriormente, essa abordagem é baseada no fato de que as partículas não necessitam seguir sempre um único regime. O principal objetivo é evitar que a população de indivíduos convirja precocemente para uma região contendo apenas mínimos locais (KROHLING; MENDEL; CAMPOS, 2011).

Com o intuito de verificar se o uso de mapas caóticos melhora o desempenho dos algoritmos populacionais analisados neste trabalho, realizou-se uma investigação experimental com a adoção das mesmas funções multimodais apresentadas previamente na Tabela 4.1 da página 46.

5.4.1 O Algoritmo Híbrido

Com a adesão de mapas caóticos ao processo de geração de números aleatórios em substituição das distribuições de probabilidades, as equações (4.1) e (4.2) para geração de novas posições para partícula i são alteradas para:

$$\boldsymbol{x}_i(t+1) = \boldsymbol{p}_{best_i}(t) \cdot [1 + \eta \cdot CJ(-1,1)] \tag{5.5}$$

em que η denota o fator de escala e $CJ(-1,1)$, dentro do intervalo [-1,1], representa o número gerado de acordo com o mapa caótico. Para cada variável $j = 1,\ldots,n$ da partícula i um novo número é gerado. O valor $CJ(-1,1)$ é obtido após o mapeamento da variável gerada dentro do intervalo [0,1] pela equação (5.1), (**??**), ou (5.3) de acordo com a seguinte regra: $CJ(-1,1) = 2 \cdot CJ(0,1) - 1$.

Os algoritmos populacionais combinados com a estratégia de *jump* com o uso do mapa logístico para escapar de mínimos locais, inicialmente propostos por Krohling, Mendel e Campos (2011), são descritos no Algoritmo 5. Caso a posição de uma partícula ultrapasse os limites superiores e inferiores de cada dimensão do espaço de busca, ao invés de agir conforme padronizado pelo algoritmo PSO original, a partícula é recolocada na mesma posição do melhor resultado obtido até o momento por essa partícula, $pbest_{i,j}$, como pode ser observado no Algoritmo 5. Teste experimentais foram realizados e indicam que essa pequena modificação influencia positivamente o processo de convergência da abordagem híbrida.

A metodologia adotada para decidir qual dos dois regimes (convencional ou regime de *jump*) deve ser usado na dinâmica de movimentação das partículas em um algoritmo populacional consiste em monitorar a função objetivo de cada partícula durante um número pré-especificado de iterações (denominado *maximum stagnation interval*) e, ao final desse período, avaliar se a partícula se encontra presa a um mínimo local. O intervalo de estagnação de cada partícula é incrementado em uma unidade sempre que não houver sinais de melhorias na *fitness*. Quando esse valor atinge seu máximo (sinal de estagnação), a partícula deve "saltar" para uma nova posição no espaço de busca, isto é, a posição dessa partícula estagnada deve ser atualizada de acordo com um segundo regime dinâmico, a estratégia de *jump*.

5.4.2 Investigação Experimental usando *Bare Bones Particle Swarm*

Com o objetivo de investigar experimentalmente o uso de mapas caóticos incorporados à estratégia de *jump* para a geração de números randômicos, ao invés do uso de distribuições de probabilidades conforme descrito no Capítulo 4, foi investigada uma abordagem combinando o *Bare Bones Particle Swarm* (BBPSO) com a estratégia de *jump* caótico para escapar de mínimos locais. Os testes foram realizados usando todos os mapas caóticos discutidos, todavia somente os resultados com BBPSO são reportados (BBPSO original, BBPSO+CJLM, BBPSO+CJGM e BBPSO+CJZM). Os resultados dos experimentos com os outros algoritmos, PSO e FIPS, são semelhantes aos obtidos com

Algoritmo 5 PSO com saltos caóticos

Parâmetros de entrada: Tamanho P da população, parâmetro de escala η e o intervalo máximo de estagnação (*maximum_stagnation_interval)*.
// inicialização aleatória da população de partículas usando distribuição de probabilidade
// uniforme, em que $\underline{x}$ e $\overline{x}$ são os limites inferior e superior respectivamente.
para cada partícula i **faça**
$\boldsymbol{x}_i = \underline{x} + (\overline{x} - \underline{x}) \cdot U(0,1)$
$\boldsymbol{p}_{best_i} = \boldsymbol{x}_i$
fim para
$g_{best} = \arg\min_{i=1,\ldots,P}\{f(\boldsymbol{x}_i)\}$ // melhor indivíduo da população
repetir
para cada partícula i **faça**
se $stagnation\_interval_i \leq maximum\_stagnation\_interval$ **então**
atualizar posição $\boldsymbol{x}_i$ de acordo com (3.2) e (3.4). // PSO
ou atualizar posição $\boldsymbol{x}_i$ de acordo com (3.6) e (3.2). // FIPS
ou atualizar posição $\boldsymbol{x}_i$ de acordo com (3.7). // BBPSO
else
atualizar posição $\boldsymbol{x}_i$ de acordo com (5.1) // Mapa Logístico
ou atualizar posição $\boldsymbol{x}_i$ de acordo com (**??**) // Mapa Gaussiano
ou atualizar posição $\boldsymbol{x}_i$ de acordo com (5.3) // Mapa Zaslawskii
$stagnation\_interval_i = 0$ // reiniciar intervalo de estagnação
fim se
se $f(\boldsymbol{x}_i) < f(\boldsymbol{p}_{best_i})$ **então** // atualizar melhor posição da partícula
$\boldsymbol{p}_{best_i} = \boldsymbol{x}_i$
else // incrementar o intervalo de estagnação em uma unidade
$stagnation\_interval_i += 1$
fim se // sem melhorias da *fitness*
// atualizar a melhor partícula da vizinhança (se necessário)
para cada variável j da partícula i **faça** // limitar posição
se $x_{i,j} > \overline{x}$ **então**
$x_{i,j} = pbest_{i,j}$
fim se
se $x_{i,j} < \underline{x}$ **então**
$x_{i,j} = pbest_{i,j}$
fim se
fim para
fim para
enquanto condição de término não atingida

o BBPSO e, portanto, não serão reportados.

As funções *benchmarks* usadas nos experimentos desta seção são as mesmas funções apresentadas na Tabela 4.1 da página 46. Novamente, focou-se em funções multimodais com muitos mínimos locais. Conforme explicado anteriormente (veja Seção 4.1), essa é exatamente a motivação principal pela introdução da estratégia de *jump*: escapar de mínimos locais.

Em todos os experimentos foram usadas 50 partículas (tamanho da população). Para dificultar o processo de otimização, adotou-se novamente a ideia de inicialização assimétrica não uniforme (BRATTON; KENNEDY, 2007). Os valores do intervalo de inicialização são mostrados na Tabela 4.1. Convencionou-se usar um número máximo de iterações (1500) como condição de parada. Cada experimento é executado 50 vezes. Os fatores de escala η para cada função testada estão listados na Tabela 4.2 (página 47) e foram determinados com base na regra descrita em (KROHLING, 2005).

O valor do parâmetro *maximum_stagnation_interval*, usado para monitorar a *fitness* (isto é, se há ou não melhoria), foi ajustado para cinco, mas outros valores apropriados podem ser usados. Além do mais, outros critérios para monitoramento de estagnação das partículas podem ser usadas.

Os resultados obtidos estão sumarizados na Tabela 5.1. Para a função g_1 e g_2, o BBPSO com *jump* caótico, especialmente com as sequências logística e Gauss, superou o desempenho do BBPSO original, como pode ser observado na Figura 5.4 e na Figura 5.5 respectivamente em termos de taxa de convergência e em termos de resultados estatísticos (veja a média dos resultados na Tabela 5.1). Para as funções g_3 e g_4, o BBPSO e também o BBPSO com *jump* encontram a solução ótima. Para as funções g_5 e g_6, resultados semelhantes são obtidos para todas as versões.

Os gráficos de convergência para essas funções são mostrados nas Figuras 5.4 até 5.9. Todas as versões testadas do BBPSO com a estratégia de *jump* convergem rapidamente conforme ilustrado nos gráficos. Além disso, os gráficos de convergência evidenciam que o desempenho da versão do BBPSO com *jump* implementado usando o mapa logístico é superior ao desempenho dos demais métodos em praticamente todos os *benchmakrs*, considerando velocidade de convergência. Nota-se pela Figura 5.1 que as sequências geram, com alta probabilidade, números randômicos próximos aos dois extremos do intervalo [0;1]. Considerando que o mapa caótico é necessário justamente no regime dinâmico quando as partículas estão estagnadas ou presas em mínimos locais, essa propriedade é fundamental para geração de pontos em regiões afastadas da vizinhança local, onde não houve melhorias na *fitness*.

Tabela 5.1: Resultados de simulação para as funções *benchmarks* usando o BBPSO convencional, BBPSO+CJLM (mapa logístico), BBPSO+CJGM (mapa Gaussiano) e BBPSO+CJZM (mapa Zaslawskii). Os resultados são dados pela média de 50 execuções.

Função	Método	Média	Desvio Padrão	Mediana	Melhor	Pior	Saltos Bem Sucedidos (%)
g_1	BBPSO	-10094.1	272.3	-10082.3	-10674.5	-9371.65	n/a
	BBPSO+CJLM	-12569.5	0.0	-12569.5	-12569.5	-12569.5	3.52072
	BBPSO+CJGM	-12569.5	0.0	-12569.5	-12569.5	-12569.5	2.53431
	BBPSO+CJZM	-12569.5	0.0	-12569.5	-12569.5	-12569.5	2.65132
g_2	BBPSO	56.6927	17.9018	53.7277	33.8286	126359	n/a
	BBPSO+CJLM	0.0	0.0	0.0	0.0	0.0	5.39412
	BBPSO+CJGM	0.0	0.0	0.0	0.0	0.0	8.85291
	BBPSO+CJZM	0.0	0.0	0.0	0.0	0.0	6.06612
g_3	BBPSO	1.1865	4.5167	0.0	0.0	19.5566	n/a
	BBPSO+CJLM	0.0	0.0	0.0	0.0	0.0	9.61545
	BBPSO+CJGM	0.0	0.0	0.0	0.0	0.0	18.0046
	BBPSO+CJZM	0.0	0.0	0.0	0.0	0.0	15.6543
g_4	BBPSO	0.0109783	0.0146363	0.00985728	0.0	0.0882895	n/a
	BBPSO+CJLM	0.0	0.0	0.0	0.0	0.0	5.59098
	BBPSO+CJGM	0.0	0.0	0.0	0.0	0.0	10.3902
	BBPSO+CJZM	0.0	0.0	0.0	0.0	0.0	8.62606
g_5	BBPSO	0.0767712	0.172767	0.0	0.0	0.936053	n/a
	BBPSO+CJLM	0.00352603	0.11595	0.0	0.0	0.622785	0.237907
	BBPSO+CJGM	0.00207338	0.014661	0.0	0.0	0.103669	1.17702
	BBPSO+CJZM	0.0062221	0.0325112	0.0	0.0	0.207317	0.712219
g_6	BBPSO	0.00439495	0.00543734	0.0	0.0	0.0109874	n/a
	BBPSO+CJLM	0.00395544	0.00761221	0.0	0.0	0.0439489	7.88175
	BBPSO+CJGM	0.00351597	0.00517737	0.0	0.0	0.0109874	11.2588
	BBPSO+CJZM	0.00395544	0.00761221	0.0	0.0	0.0439489	9.53758

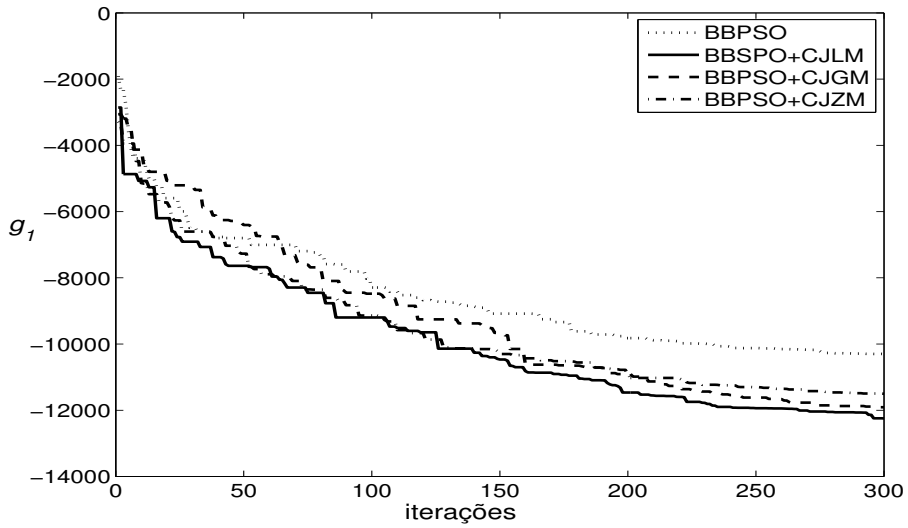


Figura 5.4: Convergência para função g_1

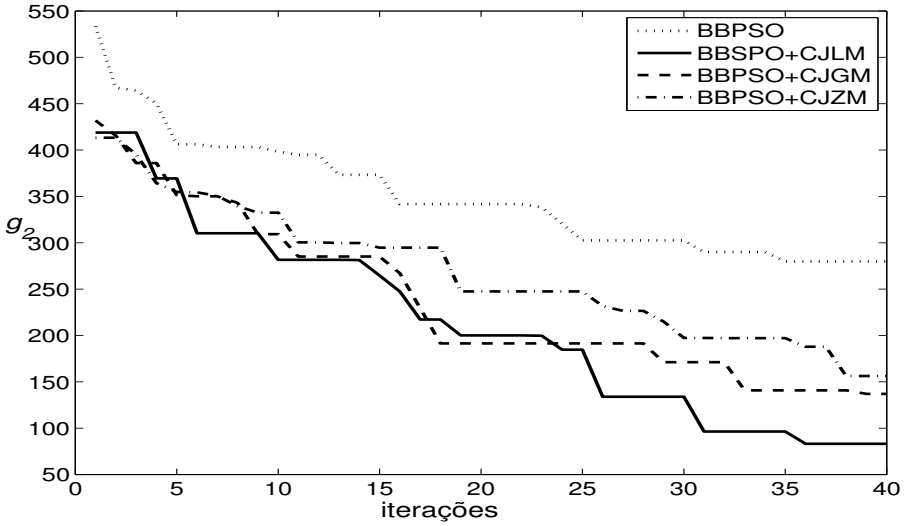


Figura 5.5: Convergência para função g_2

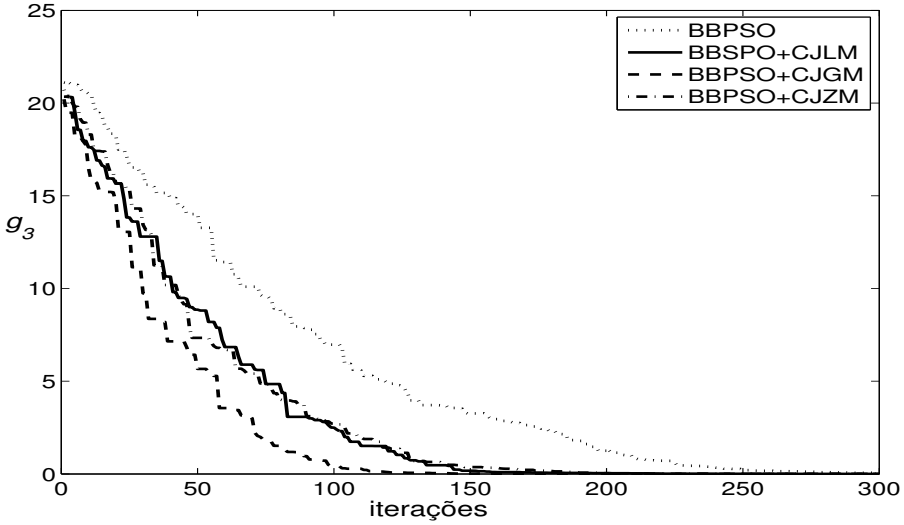


Figura 5.6: Convergência para função g_3

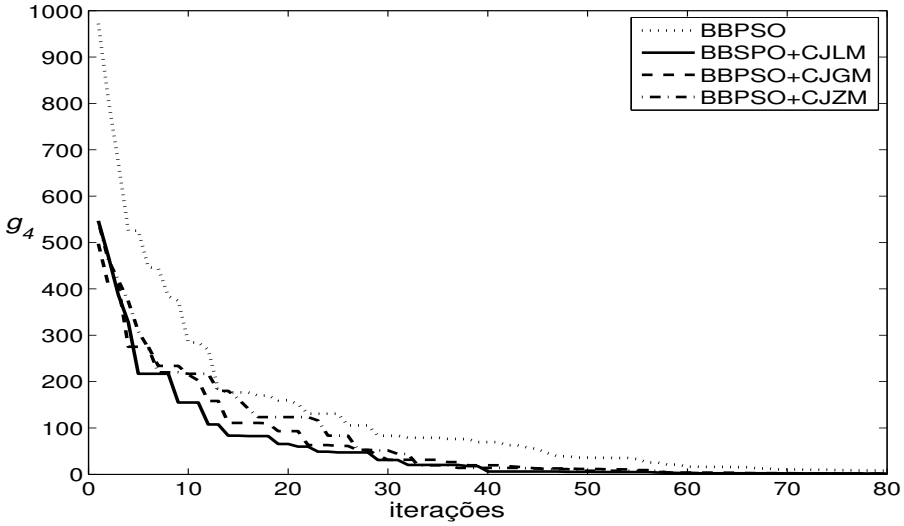


Figura 5.7: Convergência para função g_4

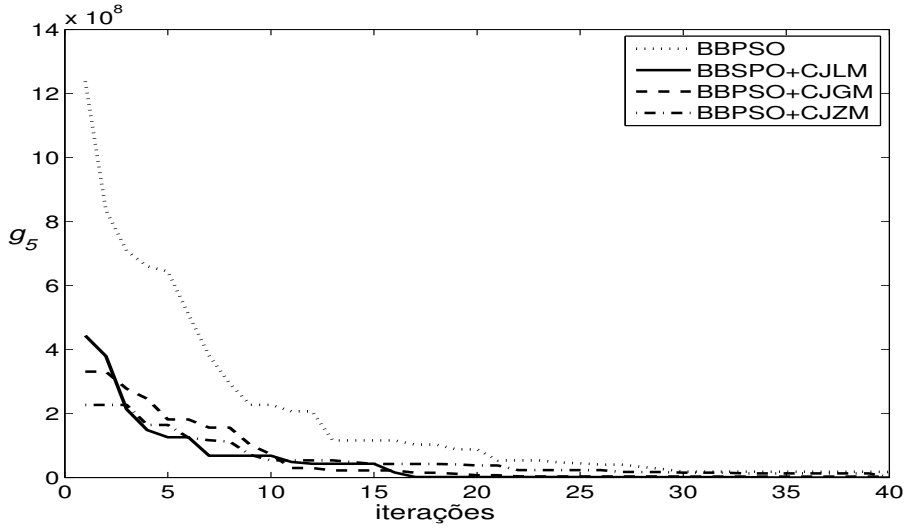


Figura 5.8: Convergência para função g_5

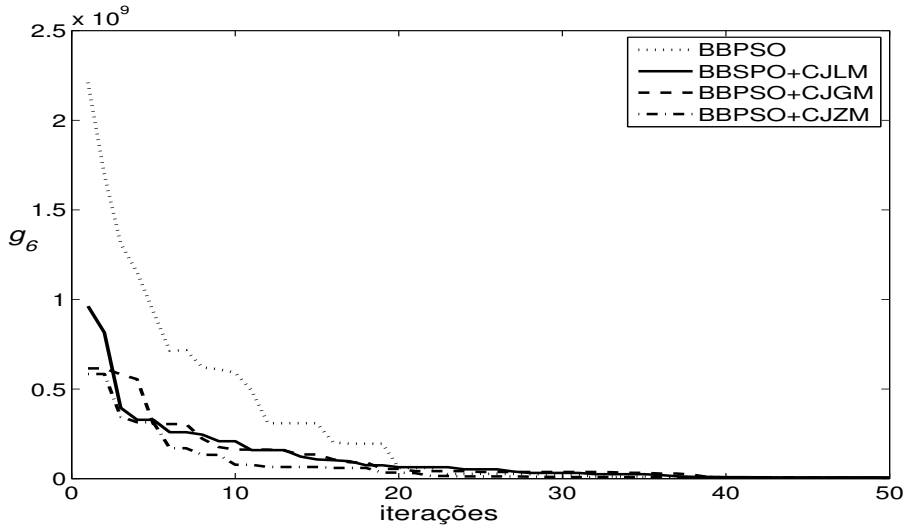


Figura 5.9: Convergência para função g_6

Em resumo, pode-se concluir que o BBPSO com *jump* caótico apresenta excelente desempenho em termos de eficácia. Isso pode ser observado na Tabela 5.1 através da pequena, porém muito importante, percentagem de *jumps* de sucesso, isto é, *jumps* que conduzem as partículas a posições promissoras no espaço de busca e, consequentemente, a melhores valores de *fitness* para todas as funções *benchmarks* multimodais investigadas. A estratégia de *jump* aumenta as chances de escapar de mínimos locais e, portanto, melhora o desempenho do algoritmo (KROHLING; MENDEL; CAMPOS, 2011). Em comparação com os resultados apresentados na Seção 4.3 (Tabela 4.3), pode-se constatar que o uso de sequência caótica no processo de otimização, incorporada à estratégia de *jump*, produz resultados melhores que os obtidos com distribuições de probabilidade. A média das execuções é melhor e o desvio padrão com caos é

inferior. Além disso, o BBPSO com *jump* caótico obtém ótimos resultados em termos de taxa de convergência. Com relação à eficiência, a geração de números randômicos usando mapas caóticos é, em média, quatro vezes mais rápida que a geração usando distribuições de probabilidades, como a Gaussiana e a Cauchy.

Os resultados de simulação apresentados nesta seção mostram que existe uma evidência estatística de que a introdução de números gerados por mapas caóticos podem impulsionar o desempenho dos algoritmos populacionais estudados em termos de exploração.

6 *Análise Experimental em Ambientes Ruidosos*

"I'm sticking in the fast lane
I never wanna quit
I'm taking all my chances
I'm gonna take 'em quick
I started at the bottom
But I'm headed to the top
Ain't nobody gonna hold me back
'Cause I'm never gonna stop."

Never Surrender
Saxon

Avaliar e comparar os desempenhos de algoritmos de otimização é uma tarefa importante porém extremamente complexa para ser conduzida analiticamente. A estratégia escolhida neste trabalho foi a de utilizar um conjunto de problemas representativos da classe de problemas comumente adotada por pesquisadores na comunidade científica, e simular ambientes ruidosos. Ambientes ruidosos são simulados computacionalmente introduzindo pequenas pertubações nas funções objetivo dos problemas analisados. Uma breve descrição de cada um dos problemas de otimização escolhidos será apresentada. Embora essas funções não forneçam precisamente uma indicação de quão eficaz será o desempenho de um algoritmo em problemas do mundo real desconhecidos, elas são úteis na investigação de aspectos importantes dos algoritmos implementados. Esta análise foi apresentada inicialmente em (MENDEL; KROHLING; CAMPOS, 2010).

Os testes experimentais foram executados considerando-se a implementação dos algoritmos tanto em suas versões originais quanto em suas versões híbridas, com a incorporação da estratégia de *jump* caótico. Desenvolvida inicialmente para ambientes

estáticos, o objetivo da estratégia de *jump* é evitar que as partículas da população sejam atraídas prematuramente para mínimos locais. Em ambientes ruidosos, a ideia subjacente à abordagem caótica de dois regimes é importante pois permite que as partículas escapem de falsos atratores criados devido à inclusão de ruído.

A estratégia de *jump* exige o ajuste de alguns parâmetros. A estimativa de um intervalo com valores aceitáveis para seleção dos parâmetros necessários à abordagem foi conduzida por meio de um estudo empírico de sensibilidade. Para o fator de escala η, o parâmetro mais sensível a variações, foram elaboradas simulações abrangentes para determinação do intervalo ótimo de valores para esse parâmetro como função do tamanho do espaço de busca de cada um dos problemas analisados. A metodologia adotada e os resultados obtidos desta análise serão detalhados neste capítulo.

Os algoritmos estudados nos experimentos deste capítulo foram implementados considerando-se apenas o mapa logístico, explicado no Capítulo 5, para geração da sequência caótica. Esse mapa, definido pela equação (5.1) apresentada na página 59, possui, adotando-se $a = 4$, excelente qualidade como gerador de números pseudo randômicos. Além disso, as trajetórias determinadas por esse mapa se comportam de maneira caótica para praticamente todos os valores pertencentes ao intervalo $[0;1]$. A decisão de incorporar o mapa logístico aos algoritmos que utilizam o regime de *jump* caótico é sustentada pelos resultados obtidos na Seção 5.4. Nos gráficos das Figuras 5.4 até 5.9 pode-se notar uma elevada taxa de convergência quando se utiliza o mapa logístico. Os valores dos demais parâmetros também são discutidos mais adiante neste capítulo.

Por último, será apresentada uma análise descritiva quantitativa dos resultados obtidos para melhor visualização da qualidade e robustez dos algoritmos híbridos em comparação com suas versões originais. Pode-se constatar uma melhoria significativa em ambos os ambientes: estático e ruidoso.

6.1 Metodologia para Simulação de Ruído

Para simular a influência de um ambiente em constante alteração e com o objetivo de analisar o desempenho de todas as versões do PSO investigadas, os valores das funções objetivos são corrompidas por ruído na seguinte maneira:

$$f_{noise}(\boldsymbol{x}) = f(\boldsymbol{x}) + N(0,\sigma^2) \tag{6.1}$$

em que $N(0,\sigma^2)$ denota uma distribuição Gaussiana com média zero e variância σ^2.

O nível de ruído (desvio padrão) é iniciado em 0.0 e incrementado até 1.0. Nos experimentos não foram utilizadas técnicas de re-amostragem, isto é, avaliar a mesma solução candidata diversas vezes para estimar o "verdadeiro" valor da *fitness* como a média dessas amostragens. Múltiplas amostragens da função objetivo reduz o erro e melhora a confiabilidade do resultado, muito embora aumente consideravelmente o custo computacional. Em um problema do mundo real, re-avaliar a mesma solução candidata pode envolver um custo tão elevado que inviabiliza essa estratégia.

6.2 Descrição das Funções de Teste (originais e corrompidas)

Neste capítulo, um conjunto de funções bem estabelecidas na literatura (LEE; YAO, 2004; MENDEL; KROHLING; CAMPOS, 2010) é utilizado para teste. Este conjunto é formado por funções *benchmarks* representantes de diversas classes de problemas. Foram consideradas funções com apenas um único mínimo, com poucos mínimos e com muitos mínimos locais para análise do desempenho dos vários algoritmos examinados. Esses *benchmarks*, numerados de f_1 até f_8, correspondem às funções: Sphere, Schaffer's F6, Ackley, Rosenbrock, Rastrigin, Griewank, Função generalizada penalizada, e Schwefel, respectivamente. Supõe-se problemas de minimização salvo se especificado o contrário. Essas funções foram escolhidas em razão de algumas particularidades, as quais transmitem grande dificuldade de otimização para a maioria das estratégias conhecidas na literatura.

O objetivo destes experimentos é investigar quais algoritmos são capazes de manter seu desempenho elevado em todas os problemas mesmo na presença de ruído. Os testes foram realizados com todos os algoritmos populacionais apresentados anterior-

mente neste trabalho. São eles: PSO canônico com topologia global, PSO padrão com topologia local em anel, FIPS e BBPSO. Um dos principais erros observados na tentativa de demonstrar as capacidades de um determinado algoritmo é o uso indevido de um conjunto de problemas cujas características não representam as dificuldades comumente encontradas no processo de otimização de funções. Ocasionalmente esse fato oculta algumas vulnerabilidades do algoritmo em questão. Por exemplo, caso nenhuma função com abundância de mínimos locais seja utilizada, é impossível mensurar a robustez de um algoritmo no combate à convergência prematura.

Nesta seção, as funções usadas na simulação serão descritas e também representadas graficamente com objetivo de visualizar a influência da adição de ruído aos cenários das funções originais. Cada descrição contém o gráfico da função original sem ruído ($\sigma^2 = 0.0$) seguido da imagem da mesma função corrompida por ruído em nível máximo ($\sigma^2 = 1.0$). Pode-se notar através dos gráficos que o ruído modifica seriamente a paisagem de várias funções, proporcionando um aumento de dificuldade no processo de otimização em virtude da inclusão de muitos falsos mínimos. Algumas poucas funções, entretanto, apenas evidenciam os efeitos da adição de informação imprecisa na região próxima ao ótimo.

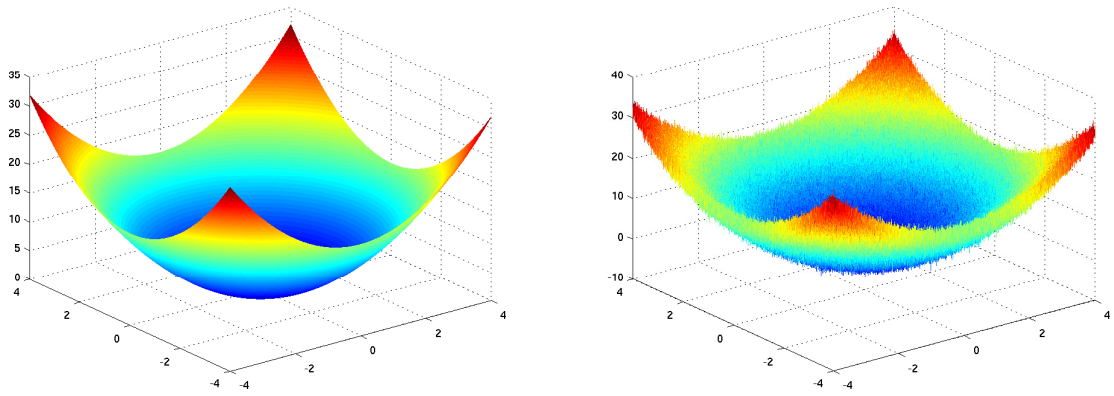
6.2.1 Sphere

Essa função é muito simples. Qualquer algoritmo de otimização numérica deveria ser capaz de solucioná-la. Sua simplicidade ajuda a realçar os efeitos da dimensionalidade nos algoritmos de otimização. Ela é descrita pela seguinte equação:

$$f_1(\boldsymbol{x}) = \sum_{i=1}^{n} {x_i}^2 \tag{6.2}$$

Dimensionalidade:	$n = 30$
Valor mínimo:	$f_{min} = 0.0$
Espaço de busca:	$(-100, 100)^n$
Espaço de inicialização assimétrica:	$(50, 100)^n$

A Figura 6.1 exibe a função Sphere original sem ruído e essa mesma função corrompida por ruído de nível $\sigma^2 = 1.0$.



(a) sem ruído ($\sigma^2 = 0.0$) (b) corrompida por ruído ($\sigma^2 = 1.0$)

Figura 6.1: Gráficos da função objetivo f_1

6.2.2 Schaffer F6

Essa é uma função complexa, com seus muitos mínimos locais organizados em círculos concêntricos ao redor do ótimo global localizado em uma depressão estreita. Algoritmos de otimização em geral ficam presos nessa região incapazes de convergir em direção ao ótimo global. A equação que representa esse problema é dado por:

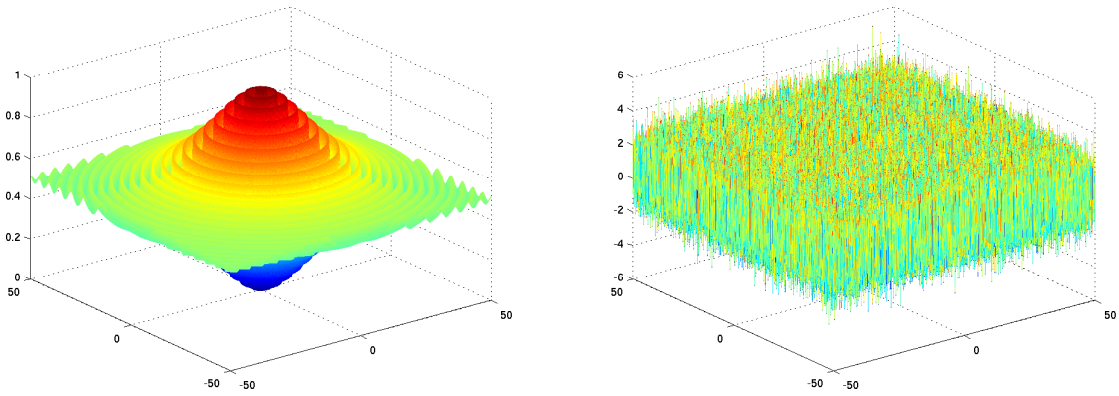
$$f_2(x_1, x_2) = 0.5 + \frac{\sin^2\left(\sqrt{x_1^2 + x_2^2}\right) - 0.5}{\left(1 + 0.001\left(x_1^2 + x_2^2\right)\right)^2} \tag{6.3}$$

Dimensionalidade:	$n = 2$
Valor mínimo:	$f_{min} = 0.0$
Espaço de busca:	$(-100, 100)^n$
Espaço de inicialização assimétrica:	$(50, 100)^n$

O formato dessa função é mostrado na Figura 6.2: (a) sem ruído e (b) com ruído.

6.2.3 Ackley

A função Ackley é um problema multimodal com vários mínimos locais. A dificuldade em otimizar esta função é decorrente da facilidade com que algoritmos de otimização são enganados pelos muitos mínimos encontrados durante a busca pelo minimizador global e induzidos a convergir prematuramente. A equação (6.4) carac-



(a) sem ruído ($\sigma^2 = 0.0$) (b) corrompida por ruído ($\sigma^2 = 1.0$)

Figura 6.2: Gráficos da função objetivo f_2

teriza essa função.

$$f_3(\boldsymbol{x}) = -20\exp\left\{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right\} - \exp\left\{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right\} + 20 + e \tag{6.4}$$

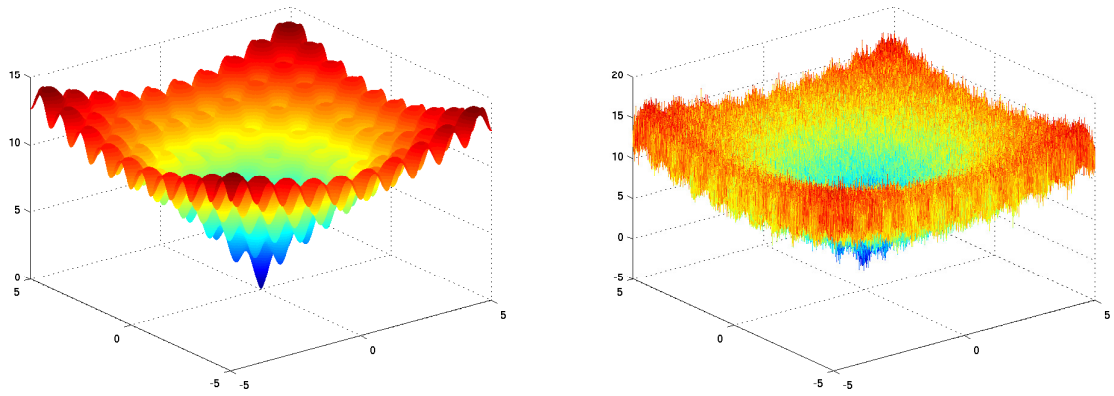
Dimensionalidade:	$n = 30$
Valor mínimo:	$f_{min} = 0.0$
Espaço de busca:	$(32,32)^n$
Espaço de inicialização assimétrica:	$(16,32)^n$

Na Figura 6.3 observa-se o quão complexa essa função é e quais são os efeitos da adição de informação imprecisa na região próxima ao ótimo. A inclusão de ruído torna o processo de otimização dessa função ainda mais complicado.

6.2.4 Rosenbrock

Nem todas as funções unimodais são simples de otimizar. A paisagem dessa função possui formato semelhante à uma banana próximo ao mínimo. A Figura 6.4 mostra como essa função é afetada pelo ruído. A variáveis da Rosenbrock são extremamente dependentes e sua função matemática é dada pela seguinte equação:

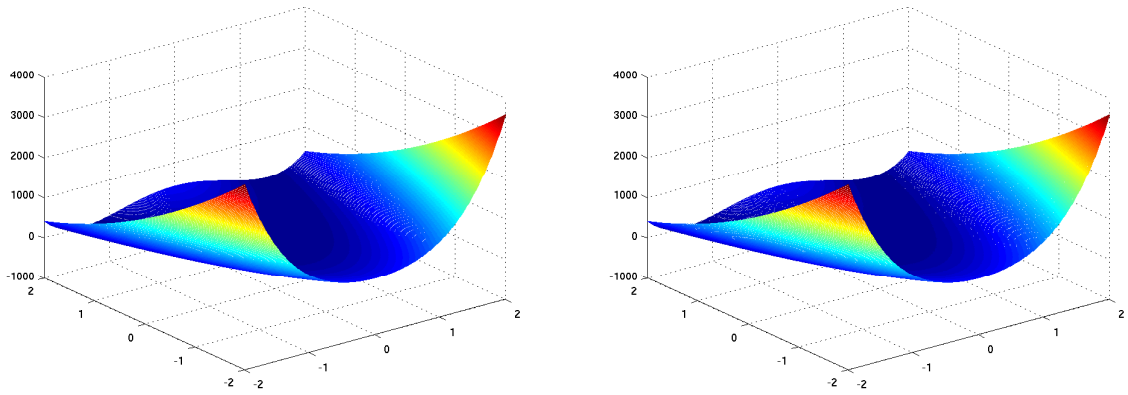
$$f_4(\boldsymbol{x}) = \sum_{i=1}^{n}\left[100\left(x_i^2 - x_{i+1}\right)^2 + (x_i - 1)^2\right] \tag{6.5}$$



(a) sem ruído ($\sigma^2 = 0.0$) (b) corrompida por ruído ($\sigma^2 = 1.0$)

Figura 6.3: Gráficos da função objetivo f_3

Dimensionalidade:	$n = 30$
Valor mínimo:	$f_{min} = 0.0$
Espaço de busca:	$(-50,50)^n$
Espaço de inicialização assimétrica:	$(25,50)^n$



(a) sem ruído ($\sigma^2 = 0.0$) (b) corrompida por ruído ($\sigma^2 = 1.0$)

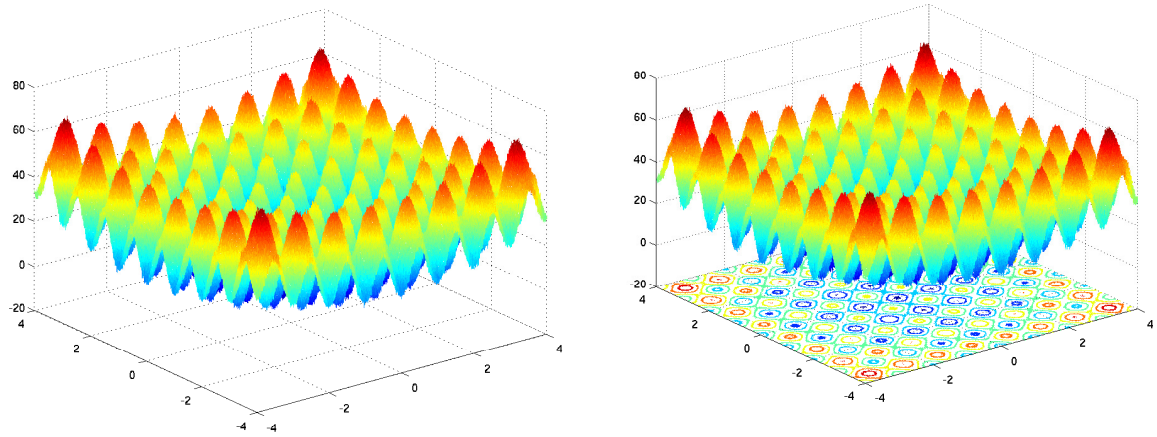
Figura 6.4: Gráficos da função objetivo f_4

6.2.5 Rastrigin

Essa é a versão multimodal da função Sphere, caracterizada por uma profunda organização de mínimos locais, como pode ser visto na Figura 6.5. Um algoritmo de otimização pode facilmente ficar preso em um mínimo local durante a busca pelo mínimo global. A equação abaixo representa a função Rastrigin:

$$f_5(\boldsymbol{x}) = \sum_{i=1}^{n} \left\{ x_i^2 - 10\cos(2\pi x_i) + 10 \right\} \tag{6.6}$$

Dimensionalidade:	$n = 30$
Valor mínimo:	$f_{min} = 0.0$
Espaço de busca:	$(5.12, 5.12)^n$
Espaço de inicialização assimétrica:	$(2.56, 5.12)^n$



(a) sem ruído ($\sigma^2 = 0.0$) (b) corrompida por ruído ($\sigma^2 = 1.0$)

Figura 6.5: Gráficos da função objetivo f_5

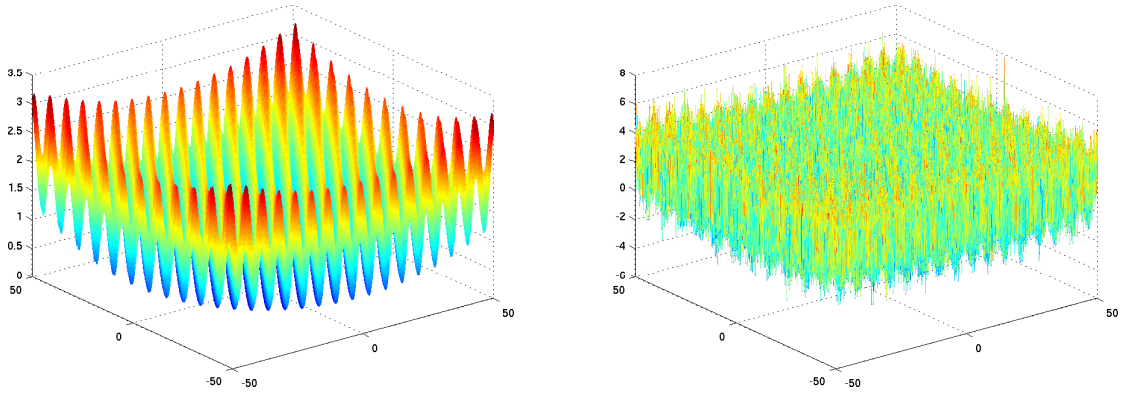
6.2.6 Griewank

Essa função é extremamente multimodal com significativa interação entre suas variáveis causada pelo termo do produto. Uma propriedade interessante dessa função é que apesar de a quantidade de mínimos locais aumentar com a dimensão, a influência do termo do produto diminui drasticamente nessas circunstâncias, tornando-se insignificante quando $n > 30$. O número de dimensões usualmente adotado na literatura é $n = 30$. A próxima equação representa a formulação matemática do problema de otimização Griewank.

$$f_6(\boldsymbol{x}) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \tag{6.7}$$

Dimensionalidade:	$n = 30$
Valor mínimo:	$f_{min} = 0.0$
Espaço de busca:	$(-600, 600)^n$
Espaço de inicialização assimétrica:	$(300, 600)^n$

A Figura 6.6 ilustra como é essa função e o quão profunda é a influência do ruído.



(a) sem ruído ($\sigma^2 = 0.0$) (b) corrompida por ruído ($\sigma^2 = 1.0$)

Figura 6.6: Gráficos da função objetivo f_6

6.2.7 Generalizada Penalizada

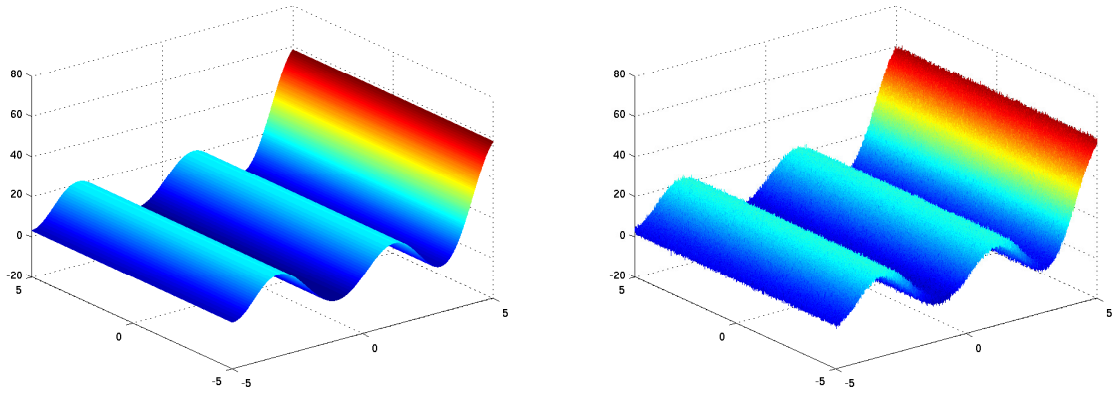
O formato dessa função pode ser visualizado na Figura 6.7.

$$f_7(\boldsymbol{x}) = \frac{\pi}{n}\left\{10\sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2\left\{1 + 10\sin^2(\pi y_{i+1})\right\} + (y_n - 1)^2\right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$$

$$y_i = 1 + (1/4) \cdot (x_i + 1)$$

$$u(x, a, k, m) = \begin{cases} k \cdot (x - a)^m, & x > a \\ 0, & -a \leq x \leq a \\ k \cdot (-x - a)^m, & x < -a \end{cases} \tag{6.8}$$

Dimensionalidade:	$n = 30$
Valor mínimo:	$f_{min} = 0.0$
Espaço de busca:	$(-50, 50)^n$
Espaço de inicialização assimétrica:	$(25, 50)^n$



(a) sem ruído ($\sigma^2 = 0.0$) (b) corrompida por ruído ($\sigma^2 = 1.0$)

Figura 6.7: Gráficos da função objetivo f_7

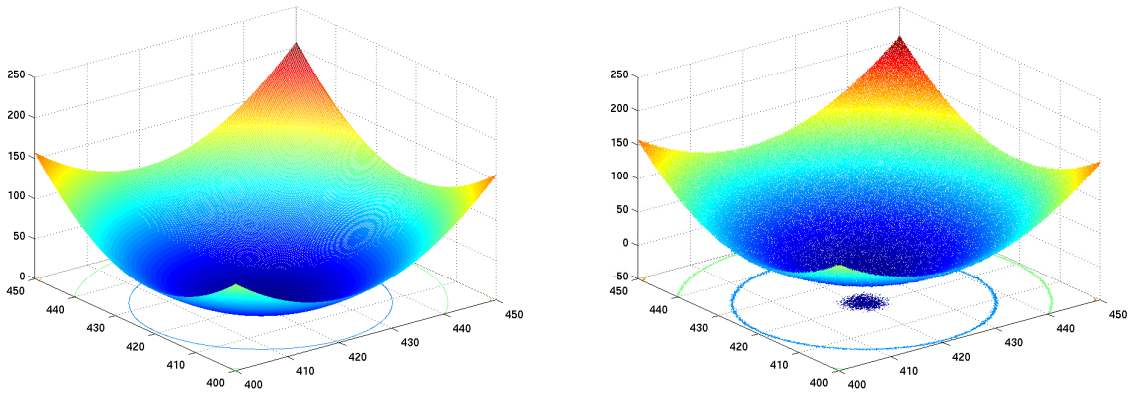
6.2.8 Schwefel

Função multimodal cujo mínimo global está localizado próximo ao canto do espaço de busca.

$$f_8(\boldsymbol{x}) = 418.9829 \cdot n - \sum_{i=1}^{n} x_i \sin(\sqrt{x_i}) \tag{6.9}$$

Dimensionalidade:	$n = 30$
Valor mínimo:	$f_{min} = 0.0$
Espaço de busca:	$(-500, 500)^n$
Espaço de inicialização assimétrica:	$(-500, -250)^n$

A Figura 6.8(a) representa a função Schwefel original enquanto a Figura 6.8(b) mostra a mesma função porém afetada por ruído.



(a) sem ruído ($\sigma^2 = 0.0$) (b) corrompida por ruído ($\sigma^2 = 1.0$)

Figura 6.8: Gráficos da função objetivo f_8

6.3 Estudo de Sensitividade

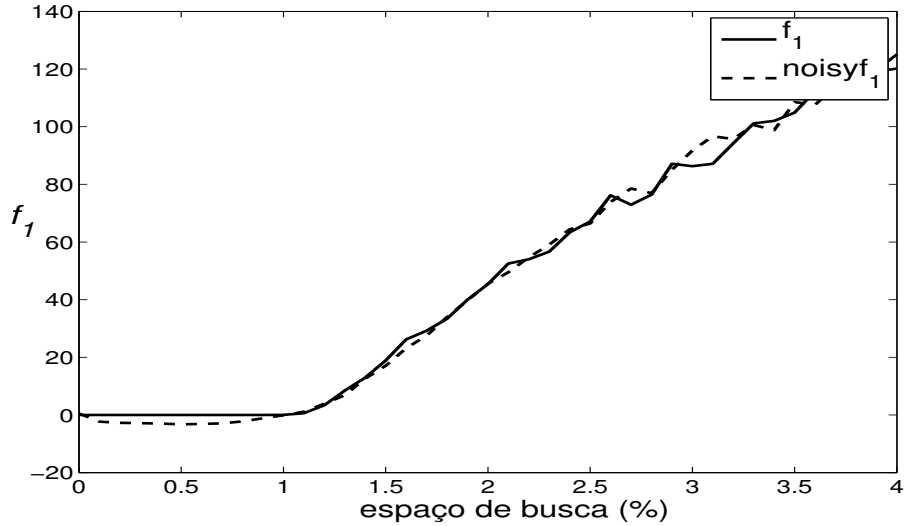
Análise sensitiva é uma técnica muito utilizada para alterar sistematicamente os parâmetros de um modelo e determinar os efeitos dessas alterações. A compreensão de como o modelo responde às variações em seus parâmetros de entrada é de fundamental importância para assegurar seu uso correto. Algumas vezes a análise de sensitividade pode revelar *insights* surpreendentes sobre o tema de interesse. Isto é, essa análise pode expor questões não previstas no início do estudo, proporcionando grande melhoria no desempenho do modelo final.

Um estudo de sensitividade foi conduzido utilizando-se o PSO canônico com o objetivo de encontrar intervalos ótimos para o fator de escala η, necessário à estratégia de *jump*, como função do tamanho do espaço de busca de cada benchmark testado. O algoritmo foi executado 50 vezes e os valores da média das funções objetivos para cada um dos valores do parâmetro η são reportados. Esses valores são expressados como percentagem do espaço de busca.

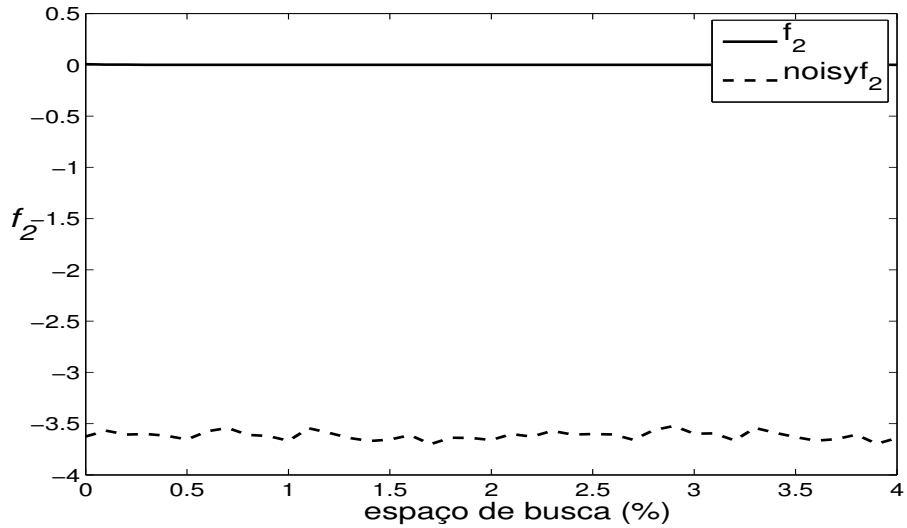
Os resultados dessa investigação estão ilustrados nos gráficos 6.9(a) até 6.9(h). Com base nessas figuras, pode-se inferir que não existe um único valor ótimo de η para todas as funções. Esse é um resultado previsível visto que cada função objetivo possui suas próprias características e quantidades de mínimos locais. Entretanto, os resultados sugerem que os valores ótimos de η pertencem ao intervalo comum $[0.1\%; 10\%]$ (MENDEL; KROHLING; CAMPOS, 2010).

Apesar desse estudo de sensitividade ter sido conduzido investigando-se somente

o PSO canônico, os resultados obtidos com os demais métodos discutidos neste trabalho são todos similares aos resultados apresentados. É importante ressaltar também que o comportamento de cada método ao otimizar cada uma das funções considerando o parâmetro η é semelhante.

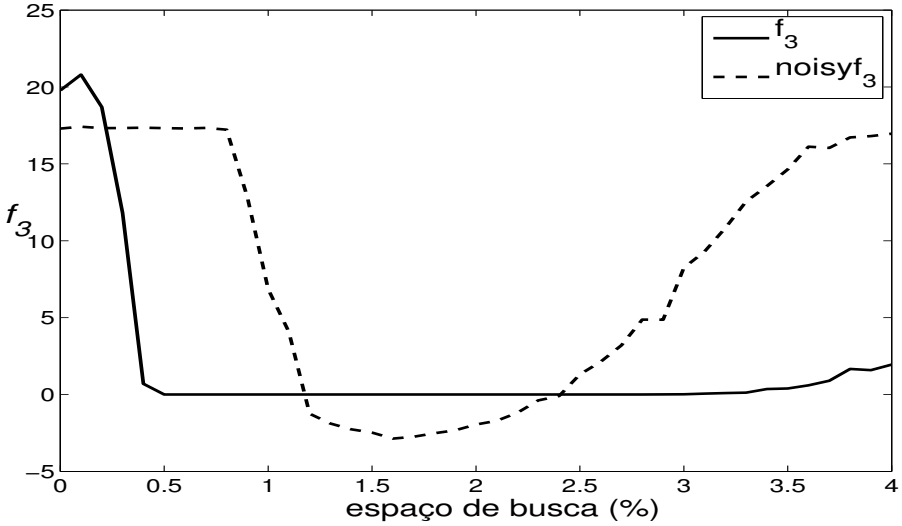


(a) Função Sphere

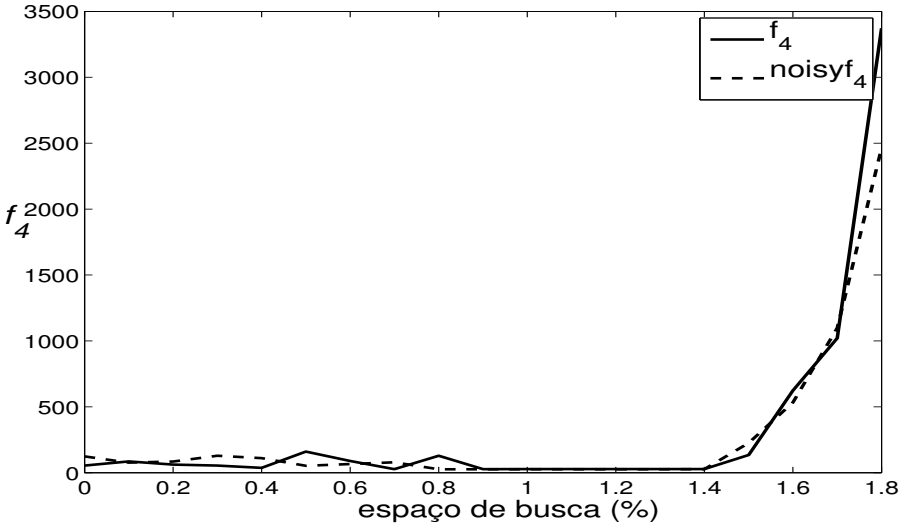


(b) Função Schaffer's F6

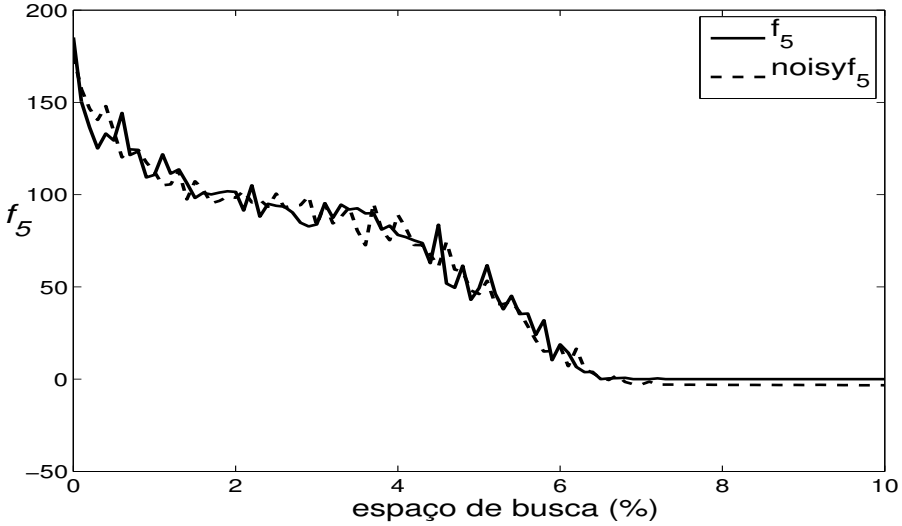
Figura 6.9: Fator de escala η como função do espaço de busca.



(c) Função Ackley

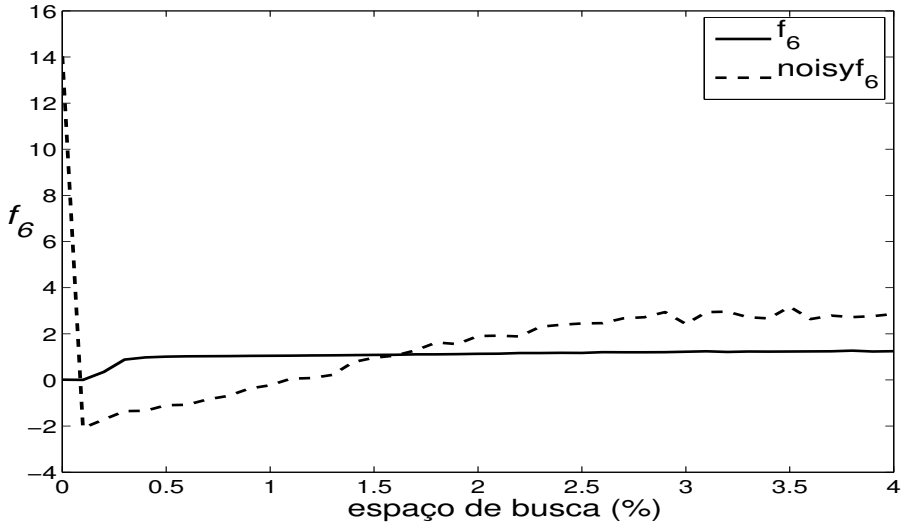


(d) Função Rosenbrock

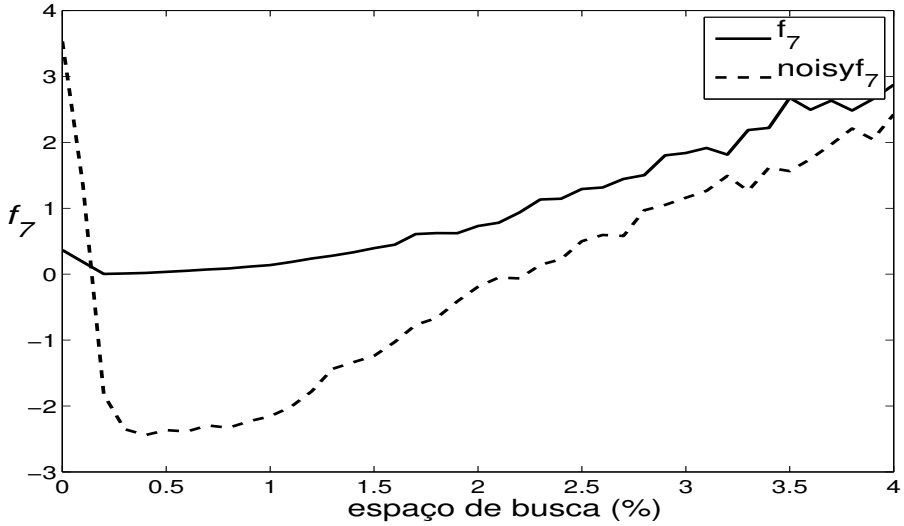


(e) Função Rastrigin

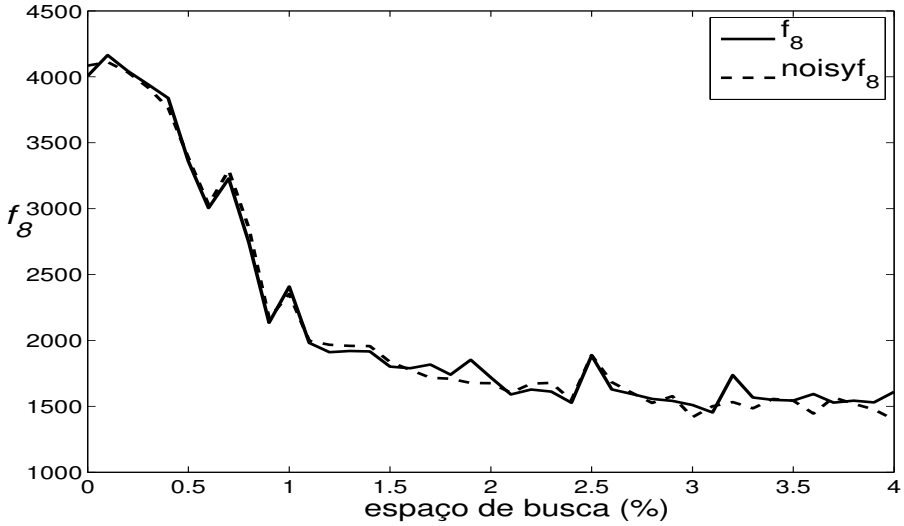
Figura 6.9: (Continuação)



(f) Função Griewank



(g) Função Generalizada Penalizada



(h) Função Schwefel

Figura 6.9: (Continuação)

Quando o parâmetro η assume valores superiores à 10% do tamanho do espaço de busca, ocorre uma degeneração do desempenho. Os valores do fator de escala podem ser obtidos para cada umas das funções a partir dos gráficos da Figura 6.9, que mostra o intervalo ótimo considerando a percentagem do espaço de busca de cada *benchmark*, escolhendo o valor desejado, considerado ótimo, e calculando seu valor real usando a equação a seguir:

$$\eta = \frac{r}{100} \cdot (\overline{x} - \underline{x}), \tag{6.10}$$

sendo que r é percentual do espaço de busca associado à função em questão, e $\overline{x}$ e $\underline{x}$ são os limites superior e inferior, respectivamente.

O desempenho dos métodos é fortemente influenciado por seus parâmetros de controle. Escolhas erradas para seus valores podem resultar em trajetórias cíclicas ou divergentes. Embora sejam fornecidas recomendações para os valores de alguns parâmetros, baseadas em estudos empíricos, esses valores não são universalmente aplicáveis. Os melhores valores continuam sendo dependentes de cada problema, e necessitam de ajuste fino para cada problema.

6.4 Configuração dos Experimentos

Esta seção contém os valores de todos os parâmetros usados nos experimentos e também específica como os testes foram realizados. Foram usados os quatro algoritmos discutidos no trabalho: PSO gbest, PSO lbest, FIPS e BBPSO. O objetivo dos experimentos é analisar o desempenho dos vários métodos testados e também da abordagem de *jump* estudada em ambientes ruidosos. Deseja-se investigar a robustez dos algoritmos, ou seja, investigar como os algoritmos se comportam à medida em que o nível de ruído aumenta. Espera-se que o uso da estratégia de *jump* evite uma eventual degradação do desempenho dos métodos de otimização. A principal intenção ao adotar esses algoritmos, bastante conhecidos na literatura, é a de investigar a influência da topologia e se é necessário manter a informação global na presença de ruído. Além disso, BBPSO é investigado pois trata-se de um método que utiliza informação global porém de maneira amostrada. Primeiramente foram realizados experimentos com os algoritmos em suas versões originais e posteriormente com suas versões melhoradas usando a estratégia de *jump*.

Com base nos resultados apresentados no Capítulo 5 optou-se por implementar a estratégia de *jump* utilizando o mapa caótico logístico. Para o caso em que $a = 4$, o mapa possui qualidade bastante elevada como gerador de números pseudo randômicos, e as trajetórias determinadas pelo mapa se comportam de maneira caótica para praticamente todos os valores $z_k \in [0;1]$. O mapa logístico com $a = 4$ gera números aleatórios com média 0.5 e desvio padrão 0.35. A Figura 5.1 (página 60) ilustra o histograma para o gerador de sequência caótica usando o mapa logístico. Pode-se observar pelo gráfico que a maioria dos números gerados pelo mapa logístico produzidos próximos aos dois extremos do intervalo $[0;1]$. Em razão do mapa caótico ser utilizado na dinâmica de dois regimes exatamente quando as partículas estão estagnadas ou presas em um mínimo local, essa característica é interessante para gerar novos pontos no espaço de busca afastados da região na qual não ocorrem melhorias da função objetivo.

Com o objetivo de aliviar o viés existente em que algoritmo tende a convergir rapidamente em direção a área do espaço onde a população foi inicializada, empregou-se novamente metodologia citada em (BRATTON; KENNEDY, 2007) cujo intervalo de inicialização assimétrico não contém o mínimo global, como indicado na Seção 6.2. Esse método força a população a expandir sua busca para além de seus limites iniciais e foi convencionado como sendo um padrão de pesquisa para comparação de

algoritmos e análise de desempenho.

Cada experimento foi executado 50 vezes e nos casos em que o valor obtido nas simulações for inferior a 10^{-8} seu valor é arredondado para 0.0. A seguir são listados os valores do parâmetros usados.

Coeficiente de aceleração O valor de φ_1 e φ_2 foi ajustado para 2.05, valor comumente adotado pela comunidade de *particle swarm*.

Coeficiente de constrição Como sugerido por (CLERC; KENNEDY, 2002), esse parâmetro foi configurado para 0.729.

Tamanho da população Em todos os experimentos deste capítulo o número de partículas (tamanho da população) foi ajustado para 20, por motivo de compatibilidade de todos os algoritmos com a estrutura topológica utilizada no método FIPS. Segundo (BRATTON; KENNEDY, 2007), o algoritmo PSO produz bons resultados usando de 20 a 100 partículas na população. Certamente o tamanho ótimo da população depende do problema em questão. Todavia está fora do escopo desse trabalho investigar qual a quantidade ótima de partículas.

Número máximo de iterações Foram utilizadas 1500 iterações e cada execução é terminada somente quando esse número máximo de iterações é alcançado.

Intervalo máximo de estagnação Esse parâmetro monitora se há ou não melhoria na *fitness* e seu valor foi ajustado para cinco. Contudo outros valores apropriados poderiam ser utilizado. Quando não há melhoria da *fitness* durante cinco iterações é um sinal de estagnação e a partícula deve mover (ou amostrar o espaço de busca) de acordo com outro regime dinâmico. Certamente, outros critérios para monitoramento da função objetivo pode ser usado.

Fator de escala Os valores dos parâmetros η (fator de escala) foram escolhidos segundo os gráficos retratados na Seção 6.3 e estão listados na Tabela 6.1. Os valores de η nos experimentos são os mesmos para todos os métodos.

6.5 Resultados e Discussões

Inicialmente foram realizados experimentos com as quatro versões dos algoritmos originais (PSOgbest, PSOlbest, FIPS, and BBPSO) otimizando funções ruidosas. Em

Tabela 6.1: Valores de η utilizados nos experimentos.

Função	Parâmetro de escala η
f_1	1.1
f_2	1.1
f_3	1.1
f_4	1.1
f_5	1.1
f_6	1.1
f_7	0.1
f_8	20

cada teste os algoritmos são executados 50 vezes e o resultado final é dado pela média das execuções. Observando-se os resultados iniciais, constata-se que nenhuma versão apresenta desempenho superior às demais versões investigadas.

A segunda etapa dos testes foi realizada usando a abordagem híbrida, ou seja, os mesmos algoritmos porém combinados com a estratégia de *jump* caótico implementado com o mapa logístico. O objetivo é analisar se os bons resultados obtidos com os algoritmos híbridos em ambiente estático se repetem na presença de ruído. Os valores usados para o parâmetro η (fator de escala) para cada uma das funções testadas estão listadas na Tabela 6.1 e foram investigados na Seção 6.3. Apesar de o mapa logístico ter sido o escolhido, nenhuma afirmação é feita à respeito de essa ser a decisão ótima. Certamente há outras boas possibilidades.

A estratégia de *jump* adicionada aos algoritmos populacionais demonstrou ser eficaz quando aplicada tanto às funções estáticas quanto às funções ruidosas pois aumentam as chances das partículas escaparem de mínimos locais. A implementação dessa abordagem é bastante simples e não eleva os custos computacionais. O uso de um gerador de sequências caóticas como alternativa às distribuições de probabilidade é responsável por manter a abordagem eficiente e eficaz. Em termos de eficiência computacional, o gerador de números pseudo randômicos usando o mapa logístico é, em média, quatro vezes mais rápido do que um gerador usando distribuição de probabilidade Gaussiana ou de Cauchy, por exemplo.

As Tabelas 6.2 até 6.5 resumem os experimentos. Cada uma delas apresenta os resultados obtidos com a versão original do método e também com sua versão híbrida. Conforme mencionado anteriormente, nenhuma das quatro versões originais (PSO com topologia global, PSO com topologia local em anel, FIPS e BBPSO) teve êxito ao

otimizar funções ruidosas. As versões modificadas se mostraram muito mais robustas e não sofreram tanto com o aumento do nível do ruído quanto as versões originais. Além disso, em ambiente estático, isto é, com nível de ruído $\sigma^2 = 0.0$, o desempenho das versões híbridas foi superior ao desempenho exibido pelos algoritmos originais na maioria das funções.

Os resultados numéricos dos testes com as versões originais dos algoritmos e também com suas versões aperfeiçoadas com *jump* caótico estão contidos, conforme mencionado anteriormente, nas tabelas 6.2 até 6.5. Essas tabelas mostram os resultados em termos da média $\pm$ desvio padrão, para cada nível de ruído especificado σ^2: 0.0 (sem ruído), 0.20, 0.40, 0.60, 0.80 e 1.0. Verifica-se que, para o caso sem ruído ($\sigma^2 = 0.0$), todas as quatro versões híbridas dos algoritmos populacionais apresentaram desempenhos semelhantes considerando as funções f_1 e f_2, visto que essas duas funções são unimodais e a estratégia de *jump* não melhora mas também não deteriora o desempenho dos algoritmos nesses casos. Para todas as demais funções multimodais sem ruído, as versões aperfeiçoadas superaram os algoritmos originais. Quando as partículas estão estagnadas a abordagem proposta possibilita que as partículas "saltem" para pontos mais promissores no espaço de busca.

Como é sabido da literatura (BRATTON; KENNEDY, 2007), as versões com topologia global do PSO (PSOgbest e BBPSO) podem exibir resultados inferiores ao otimizar funções multimodais, como é o caso das funções f_3 até f_8. Observa-se que os experimentos realizados indicam que a adição do *jump* caótico aperfeiçoa de modo similar todas as versões do PSO, entretanto o BBPSO+CJ se comporta de maneira muito robusta nas situações em que as funções são corrompidas por ruído, especialmente para a função f_8. Esse comportamento sugere que amostrar o espaço de busca ao invés de avançar como normalmente é feito no PSO, pode ser um caminho promissor para futuras investigações.

Além dos testes realizados com as quatro versões citadas, também foram feitos experimentos considerando o algoritmo FIPS com topologia dinâmica. O objetivo é analisar se uma dinâmica na topologia, que propicia uma maior aleatoriedade ao algoritmo FIPS, pode proporcionar alguma melhoria no desempenho desse método em ambientes ruidosos. A dinâmica implementada foi baseada na topologia de vizinhança aleatória discutida em (SUGANTHAN, 1999; MOHAIS et al., 2005; AKAT; GAZI, 2008), com pequenas modificações. A metodologia considera que é necessário alterar a vizinhança de uma determinada partícula somente quando não há melhoria

Tabela 6.2: Resultados experimentais usando PSO canônico (modelo *gbest*).

Função	Método	Nível de ruído (σ^2)					
		0.0	0.2	0.4	0.6	0.8	1.0
f_1	*PSOgbest*	0.0	-0.0279354	-0.0176489	-0.191759	-0.232338	-0.452971
		(0.0)	(0.354533)	(0.794596)	(1.06018)	(1.46177)	(1.17831)
	PSOgbest + *CJ*	0.0	-0.6179	-1.2300	-1.9386	-2.5353	-3.0734
		(0.0)	(0.0782)	(0.1627)	(0.2384)	(0.2833)	(0.3496)
f_2	*PSOgbest*	0.00599113	-0.308344	-1.18361	-1.98474	-2.7862	-3.6513
		(0.00660816)	(0.057603)	(0.110425)	(0.176884)	(0.20186)	(0.308736)
	PSOgbest + *CJ*	0.0	-0.6999	-1.2692	-2.0194	-2.8343	-3.6902
		(0.0)	(0.0710)	(0.1359)	(0.1720)	(0.2656)	(0.3628)
f_3	*PSOgbest*	19.7858	19.8074	19.5815	18.9115	18.1677	17.3117
		(0.046601)	(0.23214)	(0.184933)	(0.178122)	(0.19033)	(0.311449)
	PSOgbest + *CJ*	0.0	-0.5807	-1.1607	-1.7524	-2.2133	-2.5561
		(0.0)	(0.0759)	(0.1457)	(0.3129)	(0.3881)	(0.4393)
f_4	*PSOgbest*	68.1306	111.216	142.097	223.404	272.794	207.449
		(63.8278)	(148.771)	(168.987)	(347.302)	(566.667)	(221.806)
	PSOgbest + *CJ*	27.4331	27.6466	27.4420	27.0004	26.4342	25.8342
		(0.2370)	(0.4012)	(0.2429)	(0.2337)	(0.3073)	(0.4624)
f_5	*PSOgbest*	159.113	163.715	175.812	163.347	173.91	177.388
		(39.5546)	(42.9225)	(41.8345)	(39.187)	(37.7134)	(42.8399)
	PSOgbest + *CJ*	0.0	-0.6274	-1.3067	-1.8859	-2.5764	-3.2350
		(0.0)	(0.0649)	(0.1501)	(0.2160)	(0.2606)	(0.3722)
f_6	*PSOgbest*	0.0136098	1.34759	2.66158	6.33953	9.25732	14.0817
		(0.0159131)	(0.475466)	(1.17307)	(2.98258)	(3.56785)	(5.93847)
	PSOgbest + *CJ*	0.0	0.1283	-0.2824	-0.8822	-1.5765	-2.2123
		(0.0)	(0.3860)	(0.1786)	(0.2561)	(0.3016)	(0.3783)
f_7	*PSOgbest*	0.546893	3.85811	3.62172	3.72854	4.15444	3.1985
		(0.765697)	(2.041)	(2.06379)	(2.26512)	(3.01234)	(2.7044)
	PSOgbest + *CJ*	0.3411	3.0367	2.9349	2.7944	2.2475	2.3502
		(0.6216)	(1.7453)	(2.2480)	(1.7539)	(1.9541)	(2.0656)
f_8	*PSOgbest*	-4051.1	-4082.6	-4042.9	-4054.7	-4027.0	-4068.8
		(271.232)	(313.836)	(228.636)	(269.853)	(260.707)	(266.478)
	PSOgbest + *CJ*	-1705.6	-1714.5	-1725.4	-1655.0	-1743.5	-1667.4
		(344.2700)	(381.3010)	(361.3810)	(357.3190)	(367.7170)	(343.8520)

Tabela 6.3: Resultados experimentais usando PSO padrão (modelo *lbest* com topologia em anel).

Função	Método	Nível de ruído (σ^2)					
		0.0	0.2	0.4	0.6	0.8	1.0
f_1	*PSOlbest*	0.0 (0.0)	-0.351915 (0.122271)	-0.742899 (0.18863)	-1.09601 (0.279134)	-1.50022 (0.33107)	-1.92362 (0.470301)
	$PSOlbest + CJ$	0.0 (0.0)	-0.6625 (0.0730)	-1.3552 (0.1285)	-2.0004 (0.2378)	-2.6887 (0.3139)	-3.2299 (0.2887)
f_2	*PSOlbest*	0.00446982 (0.00489108)	-0.33819 (0.0716734)	-1.13077 (0.123027)	-1.99348 (0.195053)	-2.82753 (0.273902)	-3.66776 (0.254686)
	$PSOlbest + CJ$	0.0 (0.0)	-0.7557 (0.0740)	-1.3396 (0.1405)	-2.0578 (0.1789)	-2.9047 (0.2636)	-3.6582 (0.2970)
f_3	*PSOlbest*	19.767 (0.108375)	19.9756 (0.144057)	19.5604 (0.150516)	18.8611 (0.201949)	18.1527 (0.265943)	17.3001 (0.343383)
	$PSOlbest + CJ$	0.0 (0.0)	-0.6359 (0.0832)	-1.2306 (0.1440)	-1.8383 (0.2321)	-2.1987 (0.3945)	-2.5360 (0.6252)
f_4	*PSOlbest*	137.892 (155.997)	143.451 (150.708)	144.42 (165.141)	108.679 (94.8461)	156.193 (149.06)	118.338 (125.137)
	$PSOlbest + CJ$	53.2977 (84.6511)	62.9951 (59.7547)	79.5707 (96.7095)	53.4115 (68.0140)	68.6313 (57.3221)	65.6186 (55.2104)
f_5	*PSOlbest*	114.451 (21.8378)	117.625 (25.9938)	120.839 (33.1615)	129.014 (33.319)	123.045 (22.6614)	123.651 (29.2859)
	$PSOlbest + CJ$	0.0 (0.0)	-0.7090 (0.0715)	-1.4518 (0.1523)	-2.1345 (0.1730)	-2.9002 (0.2542)	-3.6628 (0.2977)
f_6	*PSOlbest*	0.0046741 (0.00957196)	0.662779 (0.116875)	0.62658 (0.313068)	1.11073 (0.642347)	1.70594 (0.948301)	2.67704 (1.46436)
	$PSOlbest + CJ$	0.0 (0.0)	0.2732 (0.2943)	-0.2779 (0.1522)	-0.9373 (0.2311)	-1.4891 (0.3031)	-2.2869 (0.3647)
f_7	*PSOlbest*	0.0889302 (0.0399541)	1.73883 (1.42671)	1.98332 (1.80089)	1.91974 (1.73451)	1.71701 (1.84657)	1.74139 (1.79168)
	$PSOlbest + CJ$	0.1634 (0.3301)	1.8383 (1.6079)	2.1477 (1.7765)	1.7286 (1.6555)	1.6468 (1.7605)	1.2028 (2.0013)
f_8	*PSOlbest*	-3697 (167.411)	-3682,43 (152.564)	-3695,18 (177.843)	-3707,98 (128.841)	-3738,46 (211.045)	-3724,89 (233.15)
	$PSOlbest + CJ$	-2181.2 (237.1540)	-1932.1 (227.1080)	-1958.3 (206.1160)	-1956.0 (302.7750)	-1924.5 (231.9450)	-1955.6 (216.1540)

Tabela 6.4: Resultados experimentais usando FIPS.

Função	Método	Nível de ruído (σ^2)					
		0.0	0.2	0.4	0.6	0.8	1.0
f_1	$FIPS$	0.0	-0.591747	-1.17277	-1.81504	-2.40423	-2.91408
		(0.0)	(0.0722389)	(0.134928)	(0.252471)	(0.336935)	(0.354139)
	$FIPS + CJ$	0.0	-0.6282	-1.2181	-1.8613	-2.5072	-3.1066
		(0.0)	(0.0704)	(0.1243)	(0.1832)	(0.2824)	(0.3264)
f_2	$FIPS$	0.0535764	-0.309552	-1.17636	-1.98734	-2.85191	-3.69968
		(0.101098)	(0.0448875)	(0.120637)	(0.163232)	(0.24658)	(0.297281)
	$FIPS + CJ$	0.0	-0.6703	-1.2325	-2.0065	-2.7470	-3.6202
		(0.0)	(0.0593)	(0.1314)	(0.1650)	(0.2050)	(0.3063)
f_3	$FIPS$	20.581	20.3746	19.7425	18.9222	18.1897	17.3374
		(0.168513)	(0.0967063)	(0.127022)	(0.221037)	(0.228137)	(0.297917)
	$FIPS + CJ$	0.0	-0.6214	-1.2286	-1.8811	-2.4315	-3.0266
		(0.0)	(0.0815)	(0.1621)	(0.1936)	(0.3458)	(0.3783)
f_4	$FIPS$	86.4122	92.7598	82.7514	81.5891	84.632	85.0611
		(99.7661)	(101.603)	(87.8384)	(69.491)	(101.438)	(78.1266)
	$FIPS + CJ$	27.8806	27.9793	27.6688	27.0603	26.4406	25.7551
		(0.1942)	(0.2855)	(0.1609)	(0.1879)	(0.2281)	(0.3936)
f_5	$FIPS$	49.9891	51.9401	52.519	55.0965	52.2457	51.9314
		(14.6747)	(14.6685)	(17.2782)	(15.6633)	(15.8451)	(16.2921)
	$FIPS + CJ$	0.0	-0.6774	-1.3374	-2.0412	-2.7122	-3.3016
		(0.0)	(0.0675)	(0.1379)	(0.2153)	(0.2803)	(0.3709)
f_6	$FIPS$	0.000448295	0.739649	1.32778	2.49422	5.05123	7.84846
		(0.00183006)	(0.144834)	(0.563287)	(1.27842)	(1.92945)	(2.88424)
	$FIPS + CJ$	0.0	0.2695	-0.2907	-0.8853	-1.5340	-2.2303
		(0.0)	(0.2411)	(0.1283)	(0.1937)	(0.2238)	(0.3037)
f_7	$FIPS$	0.0186599	-0.407333	-1.12325	-1.55083	-2.13386	-2.59542
		(0.0684769)	(0.464917)	(0.231858)	(0.585771)	(0.726966)	(0.77257)
	$FIPS + CJ$	0.0124	-0.4250	-0.7953	-1.0767	-1.6132	-2.0941
		(0.0340)	(0.4282)	(0.5349)	(0.8094)	(0.8857)	(0.8117)
f_8	$FIPS$	-3574.8	-3582.6	-3586.9	-3580.4	-3573.7	-3584.2
		(67.2659)	(54.5229)	(62.0888)	(62.5405)	(50.0196)	(69.3288)
	$FIPS + CJ$	-2878.4	-2870.8	-2892.4	-2936.7	-2860.9	-2915.2
		(210.09)	(205.4260)	(200.1270)	(208.9300)	(270.9190)	(205.6330)

Tabela 6.5: Resultados experimentais usando BBPSO.

Função	Método	Nível de ruído (σ^2)					
		0.0	0.2	0.4	0.6	0.8	1.0
f_1	$BBPSO$	0.0	-0.143758	-0.3360	-0.287378	-0.793702	-0.769973
		(0.0)	(0.27331)	(0.3392)	(1.66297)	(0.700648)	(1.29733)
	$BBPSO + CJ$	0.0	-0.6245	-1.2490	-1.8240	-2.5487	-3.2321
		(0.0)	(0.0675)	(0.1470)	(0.2631)	(0.3584)	(0.4206)
f_2	$BBPSO$	0.00524659	-0.451843	-1.15306	-1.95635	-2.80347	-3.62152
		(0.00489155)	(0.177816)	(0.104863)	(0.173928)	(0.262401)	(0.299499)
	$BBPSO + CJ$	0.0002	-0.7290	-1.2842	-2.0125	-2.8839	-3.5459
		(0.0014)	(0.0726)	(0.1337)	(0.1816)	(0.2373)	(0.2901)
f_3	$BBPSO$	5.16247	14.5315	17.9542	18.8083	17.9719	17.2168
		(7.72388)	(7.48877)	(4.13043)	(0.219078)	(0.238125)	(0.266086)
	$BBPSO + CJ$	0.0	-0.5852	-1.1434	-1.7523	-2.2656	-2.7586
		(0.0)	(0.0877)	(0.1602)	(0.2478)	(0.3453)	(0.4725)
f_4	$BBPSO$	43.0079	68.1499	86.2037	116.079	117.961	116.666
		(43.7738)	(40.0037)	(58.214)	(118.191)	(132.65)	(117.742)
	$BBPSO + CJ$	37.0569	51.2435	41.4108	73.3085	71.8482	79.0311
		(24.8931)	(65.9986)	(29.0796)	(122.5390)	(92.0508)	(126.7070)
f_5	$BBPSO$	88.4914	87.0962	86.9632	90.7132	85.3097	81.6083
		(22.4138)	(21.6163)	(27.1667)	(29.0696)	(28.8104)	(20.9174)
	$BBPSO + CJ$	0.0	-0.6250	-1.2786	-1.8684	-2.6042	-3.2128
		(0.0)	(0.0872)	(0.1542)	(0.2390)	(0.4216)	(0.5188)
f_6	$BBPSO$	0.013161	0.837333	0.68798	0.485665	0.319894	0.244972
		(0.0257087)	(0.192375)	(0.417729)	(0.595895)	(0.811376)	(1.22175)
	$BBPSO + CJ$	0.0	-0.2880	-0.3574	-0.8806	-1.5681	-2.1935
		(0.0)	(0.3657)	(0.4221)	(0.3880)	(0.3247)	(0.3652)
f_7	$BBPSO$	0.218228	1.49131	2.03098	1.9437	2.24818	1.69998
		(0.377021)	(0.932523)	(1.41343)	(1.34179)	(1.89115)	(1.54257)
	$BBPSO + CJ$	0.2412	1.6233	1.9022	1.6715	1.9137	1.6758
		(0.6111)	(1.6518)	(1.3500)	(1.4828)	(1.8212)	(1.7654)
f_8	$BBPSO$	-2995.1	-2967.3	-2895.0	-2908.9	-2926.8	-2908.3
		(382.28)	(296.079)	(334.693)	(288.324)	(352.94)	(268.394)
	$BBPSO + CJ$	-400.1	-423.1	-411.9	-271.3	-359.3	-425.9
		(541.2900)	(329.6320)	(330.6440)	(243.4980)	(428.9430)	(372.9750)

na *fitness* durante um período pré-definido de iterações, visto que isso é um indicativo de estagnação em decorrência da existência de falsos mínimos locais. Dessa maneira, sempre que uma partícula encontrar-se estagnada, toda a sua vizinhança deverá ser reorganizada. Os números aleatórios são gerados a partir de sequências caóticas ao invés de distribuições de probabilidades. A despeito de boas expectativas, os testes sugerem que a inclusão de mais dinamismo durante o processo de otimização na presença de ruído não proporciona um aumento na qualidade do resultado final. Esses resultados foram muito semelhantes aos obtidos anteriormente usando a estratégia convencional de topologia fixa. Por esse motivo, esses resultados não serão reportados. Está fora do escopo desse trabalho investigar com maiores detalhes como o uso de estratégias dinâmicas podem impulsionar os algoritmos em ambientes ruidosos. Contudo, notou-se que introduzir dinamismo na topologia do FIPS pode ser uma direção interessante e atraente para futuras pesquisas na área de ambientes dinâmicos.

6.6 Testes Estatísticos em Ambiente sem Ruído

Observa-se nas Tabelas 6.2, 6.3, 6.4 e 6.5 que os algoritmos populacionais combinados com a estratégia de *jump* caótico apresentam melhores valores de média que a versão correspondente sem *jump* em muitos problemas. Com a finalidade de demonstrar se os desempenhos obtidos pelas versões híbridas são estatisticamente significante em comparação com suas respectivas versões originais, foram realizados testes estatísticos de hipótese nos resultados para todas as funções f_1 até f_8. O método adotado foi o teste t não pareado bi-caudal. As Tabelas 6.6, 6.7, 6.8, 6.9 contém os resultados dos testes e as conclusões obtidas a partir de cada um. O valor t com grau de liberdade 49 é significante ao nível 0.05. IC representa o intervalo de confiança.

Em geral, a média calculada de uma amostra não é exatamente igual à média da população. O tamanho da discrepância depende do tamanho e da variabilidade (desvio padrão) da amostra. Caso esta seja pequena e variável, a média estimada pode ficar muito longe da média real. Em contrapartida, com uma amostra grande e com pouca dispersão, a média estimada estará muito próxima da média da população. Cálculos estatísticos combinam o tamanho da amostra e sua variabilidade para geração de um intervalo de confiança (IC) para a média da população. Pode-se calcular intervalos para quaisquer graus de confiança, entretanto, 95% é o mais comum. Na comparação de grupos, calcula-se o intervalo de confiança para a diferença entre as

médias dos grupos. A interpretação é direta: se a hipótese for aceita, existe 95% de chance de o intervalo calculado incluir a verdadeira diferença entre as médias das populações.

Tabela 6.6: Valores do teste t aplicado aos resultados da Tabela 6.2, para os métodos PSOgbest e PSOgbest+CJ.

Função	t	p	IC (95%)	Conclusão
f_1	-	-	-	teste não aplicável
f_2	6.41	0.0	$[0.004;0.007]$	diferença muito significativa
f_3	3002.226	0.0	$[19.772;19.798]$	diferença muito significativa
f_4	4.508	0.0	$[22.784;58.610]$	diferença muito significativa
f_5	28.444	0.0	$[148.012;170.213]$	diferença muito significativa
f_6	6.047	0.0	$[0.009;0.018]$	diferença muito significativa
f_7	1.475	0.1433	$[-0.07;0.482]$	diferença não significativa
f_8	37.841	0.0	$[-2468.501;-2222.498]$	diferença muito significativa

Tabela 6.7: Valores do teste t aplicado aos resultados da Tabela 6.3, para os métodos PSOlbest e PSOlbest+CJ.

Função	t	p	IC (95%)	Conclusão
f_1	-	-	-	teste não aplicável
f_2	6.462	0.0	[0.003;0.006]	diferença muito significativa
f_3	1289.723	0.0	[19.736;19.797]	diferença muito significativa
f_4	3.37	0.0	[34.784;134.405]	diferença significativa
f_5	37.059	0.0	[108.322;120.579]	diferença muito significativa
f_6	3.452	0.0008	[0.002;0.007]	diferença muito significativa
f_7	1.583	0.1165	[-0.167;.018]	diferença não significativa
f_8	36.922	0.0	[-1597.268;-1434.331]	diferença muito significativa

Interpretação do valor p Dadas duas médias diferentes calculadas a partir de duas amostras, deseja-se saber se a diferença entre elas é devido à técnica investigada. A simples observação das médias não é suficiente para concluir que as populações possuem médias distintas. É possível que ambas as populações tenham a mesma média, e que a diferença observada entre elas seja apenas coincidência. Não há como

Tabela 6.8: Valores do teste t aplicado aos resultados da Tabela 6.4, para os métodos FIPS e FIPS+CJ.

Função	t	p	IC (95%)	Conclusão
f_1	-	-	-	teste não aplicável
f_2	3.747	0.0003	[0.025;0.082]	diferença muito significativa
f_3	863.611	0.0	[20.534;20.628]	diferença muito significativa
f_4	4.148	0.0	[30.532;86.53]	diferença muito significativa
f_5	24.087	0.0	[45.87; 54.107]	diferença muito significativa
f_6	1.732	0.0864	[-0.000065; 0.00096]	diferença pouco significativa
f_7	0.579	0.5639	[-0.015;0.027]	diferença não significativa
f_8	22.322	0.0	[-758.309;-634.49]	diferença muito significativa

ter certeza se a diferença observada reflete uma diferença verdadeira ou é apenas coincidência de amostras aleatórias. O que se pode fazer é calcular probabilidades.

Cálculos estatísticos podem responder à seguinte questão: "Caso as populações tenham realmente a mesma média, qual a probabilidade de se observar uma diferença tão grande (ou maior) entre as médias estimadas em um experimento deste tamanho ?" A resposta a esse questionamento é determinada pelo valor p.

O valor p é uma probabilidade. Quando seu valor é pequeno, pode-se concluir que, possivelmente, a diferença entre as médias não é coincidência. Ao contrário, com um valor alto de p, conclui-se que as populações possuem a mesma média. Ao analisar o valor p, utiliza-se o conceito de hipótese nula. Este termo afirma que não há diferença entre as médias dos grupos. Desse modo, o valor p bi-caudal é definido como a probabilidade de se observar uma diferença tão grande quanto (ou maior que) a diferença calculada, assumindo que a hipótese nula seja verdadeira.

É comum equivocar-se ao interpretar os valores p. Por exemplo, se um valor p é igual à 0.03, então existe 3% de chance de observar uma diferença tão grande quanto à observada nos experimentos, ainda que ambas as populações pertençam a mesma distribuição (hipótese nula verdadeira). É tentador concluir que, por consequência, existe 97% de chance de a diferença observada refletir a diferença verdadeira entre as populações e uma chance de 3% de que essa diferença seja aleatória. Entretanto, esta é uma conclusão incorreta. Somente é possível afirmar que amostras randômicas coletadas de populações idênticas podem apresentar uma diferença inferior à observada

Tabela 6.9: Valores do teste t aplicado aos resultados da Tabela 6.5, para os métodos BBPSO e BBPSO+CJ.

Função	t	p	IC (95%)	Conclusão
f_1	-	-	-	teste não aplicável
f_2	7.014	0.0	[0.00360.0064]	diferença muito significativa
f_3	4.726	0.0	[2.995; 7.33]	diferença muito significativa
f_4	0.836	0.4054	[-8.182;20.083]	diferença não significativa
f_5	27.917	0.0	[82.201;94.781]	diferença muito significativa
f_6	3.6199	0.0	[0.006;0.02]	diferença muito significativa
f_7	0.2262	0.8215	[-0.224;0.178]	diferença não significativa
f_8	27.6901	0.0	[-2780.976;-2409.023]	diferença muito significativa

em 97% dos experimentos e superior à observada em 3% dos experimentos.

6.7 Análise Descritiva Quantitativa

Análises descritivas são usadas para descrever resultados quantitativamente e de maneira tratável e são importantes durante a análise de dados em diferentes maneiras. Esta seção apresenta uma análise descritiva da abordagem populacional estudada. O objetivo desta análise é oferecer informações úteis e uma perspectiva geral do padrão de comportamento dos algoritmos investigados, considerando tanto ambientes estáticos quanto ruidosos.

6.7.1 Taxa de Saltos Caóticos Bem-Sucedidos

Experimentos foram desenvolvidos a fim de demonstrar claramente a eficácia da estratégia de *jump*. Analisou-se somente o desempenho da versão original do PSO com topologia global e sua versão híbrida com *jump* caótico, PSOgbest+CJ. Os parâmetros foram usados do mesmo modo como explicado na Seção 6.4. As funções *benchmarks* também são as mesmas descritas anteriormente na Seção 6.2. O comportamento dos outros métodos são similares ao exibido pela versão canônica do PSO analisada, com e sem *jump*.

Conclui-se a partir das Figuras 6.10(a) e 6.10(b) que o sucesso da abordagem deve-se ao número de *jumps* bem sucedidos ocorridos durante o início do processo de

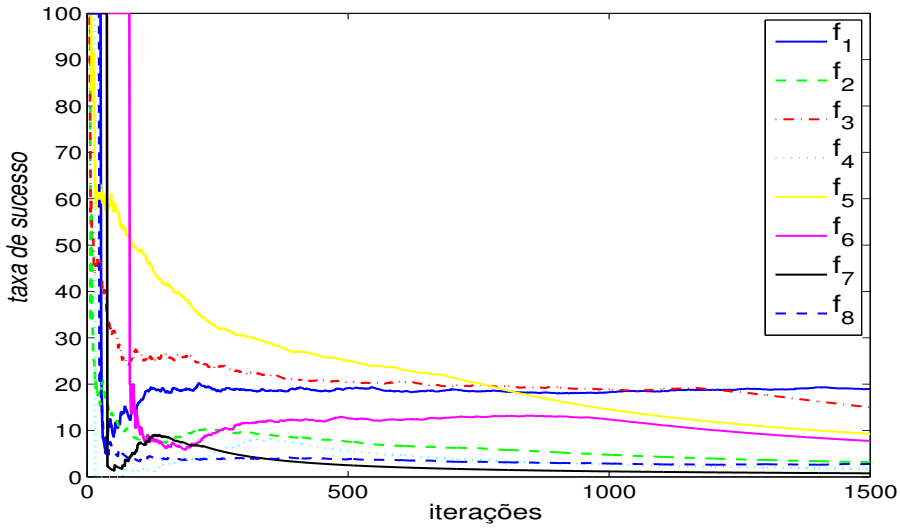
otimização. A ocorrência de *jumps* bem sucedidos no estágio inicial é desejável pois promove mais exploração do espaço de busca. Após a fase de exploração (cerca de 500 iterações nos testes realizados), os *jumps* não são mais tão efetivos visto que, como os estudos evidenciam, há uma tendência de as partículas já estarem situadas em uma região promissora. Essa característica indica a utilidade da estratégia de *jump*, ou seja, escapar de mínimos locais. É importante ressaltar que a estratégia de *jump* caótico também contribui para evitar que as partículas sejam atraídas prematuramente por falsos mínimos locais embutidos na função pela adição de ruído. A Figura 6.10(b) indica que o comportamento da abordagem híbrida ao otimizar funções corrompidas por ruído ($\sigma^2 = 1.0$) é semelhante ao comportamento exibido na ausência de incertezas.

6.7.2 Robustez dos Algoritmos

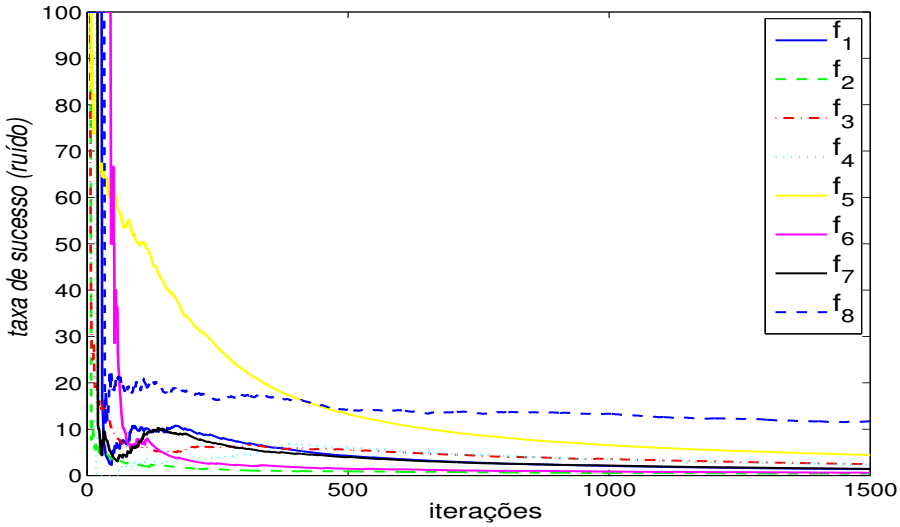
Em ambientes ruidosos, um algoritmo para ser considerado robusto precisa alcançar sistematicamente uma solução satisfatória. Não é suficiente obter bons resultados sob baixos níveis de ruídos e degradar a solução à proporção que o nível de ruído aumenta.

Análise descritiva quantitativa foi adotada a fim de se analisar a robustez dos algoritmos discutidos tanto em suas versões originais (PSOgbest, PSOlbest, BBPSO e FIPS) quanto em suas versões melhoradas com *jumps* caóticos (PSOgbest+CJ, PSOlbest+CJ, BBPSO+CJ e FIPS+CJ). As Figuras 6.11 até 6.18 mostram como a solução (média dos resultados de 50 execuções) dos algoritmos variam com o aumento do nível de ruído. Os gráficos indicam a variação na qualidade da solução com o aumento de informações incertas para as versões originais dos algoritmos e também para as versões usando *jump* caótico. Isso é importante para avaliar a robustez dos algoritmos, ou seja, como os algoritmos se comportam quando o nível de ruído adicionado às funções é aumentado. Nos casos em que não é possível visualizar satisfatoriamente todos os métodos no mesmo gráfico, as curvas foram separadas em dois gráficos distintos.

Depreende-se dos gráficos que o aumento do nível de ruído implica em uma deterioração dos resultados (média e desvio padrão) obtidos pelos algoritmos, em ambas as versões. Pode-se notar que algumas funções, como a f_2 e a f_6, tem seu desempenho deteriorado linearmente, enquanto outras apresentam um comportamento mais estável, como as funções f_5 e f_8 que mantém o mesmo declive apesar da adição de incertezas. Entretanto, essa deterioração é muito menor com a utilização da estratégia



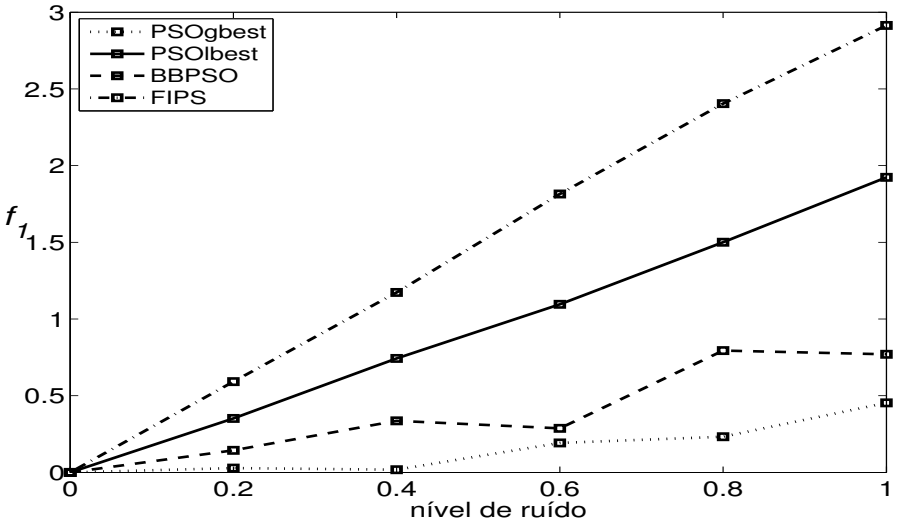
(a) Nível de ruído $\sigma^2 = 0.0$



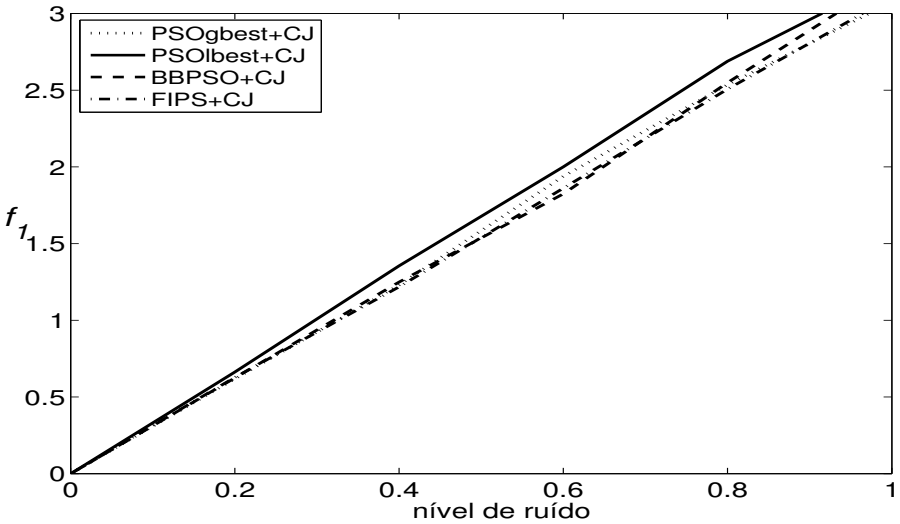
(b) Nível de ruído $\sigma^2 = 1.0$

Figura 6.10: Gráfico ilustrando o número de *jumps* bem sucedidos durante o processo de otimização.

de *jump* caótico.

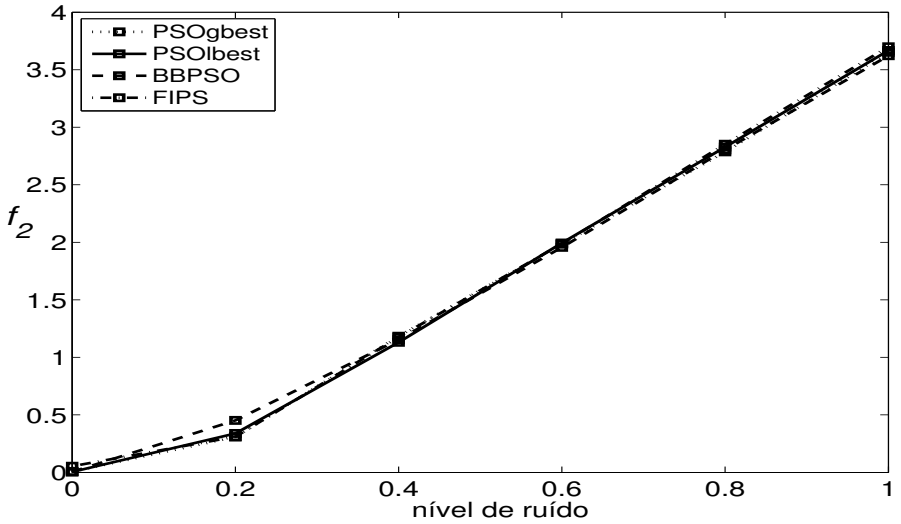


(a) Algoritmos originais

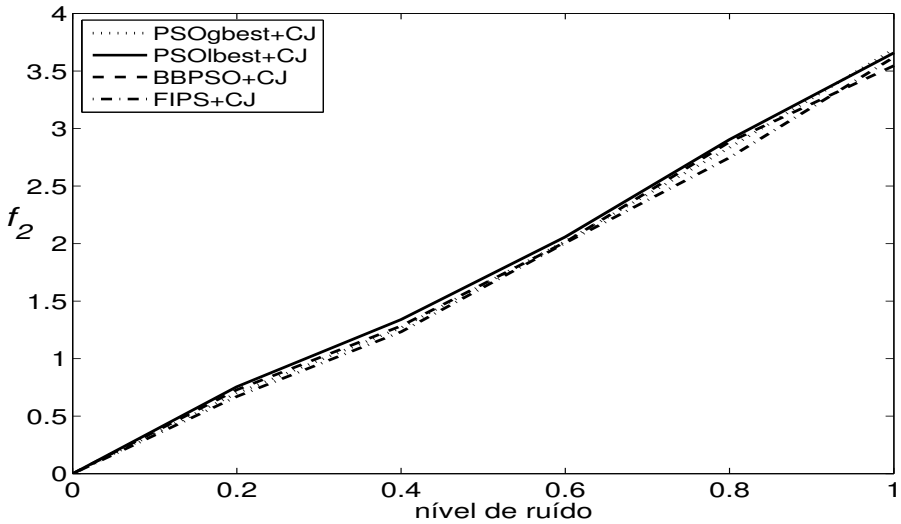


(b) Abordagens híbridas usando saltos caóticos

Figura 6.11: Variação da qualidade da solução para a função Sphere.



(a) Algoritmos originais



(b) Abordagens híbridas usando saltos caóticos

Figura 6.12: Variação da qualidade da solução para a função Schaffer F6.

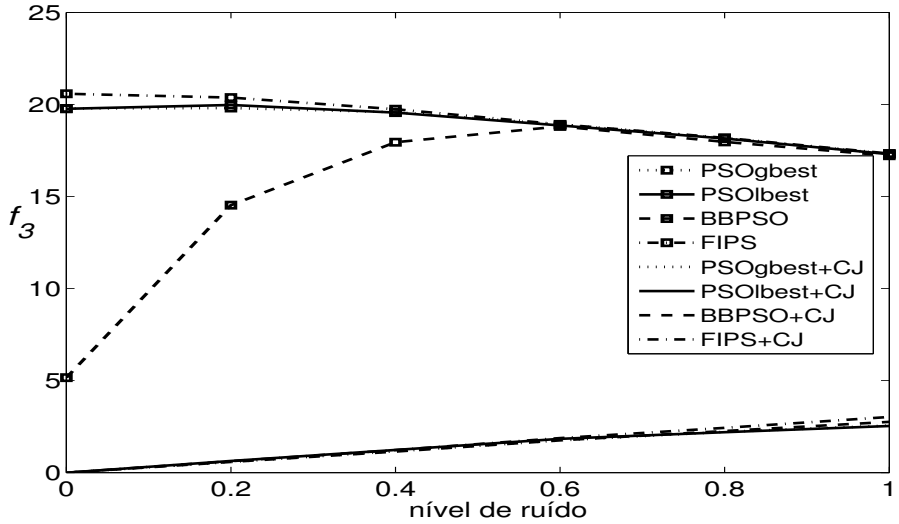


Figura 6.13: Variação da qualidade da solução para a função Ackley para ambas as versões dos algoritmos populacionais, original e híbrida com saltos caóticos.

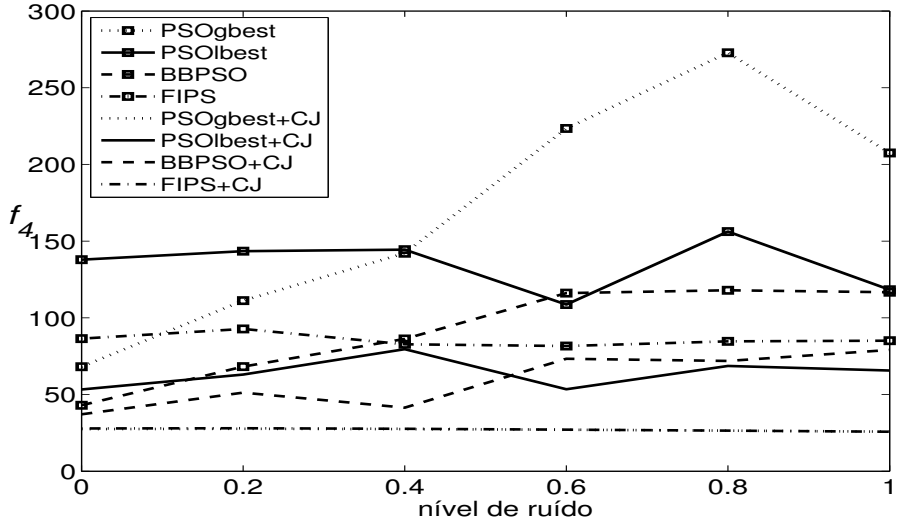


Figura 6.14: Variação da qualidade da solução para a função Rosenbrock para ambas as versões dos algoritmos populacionais, original e híbrida com saltos caóticos.

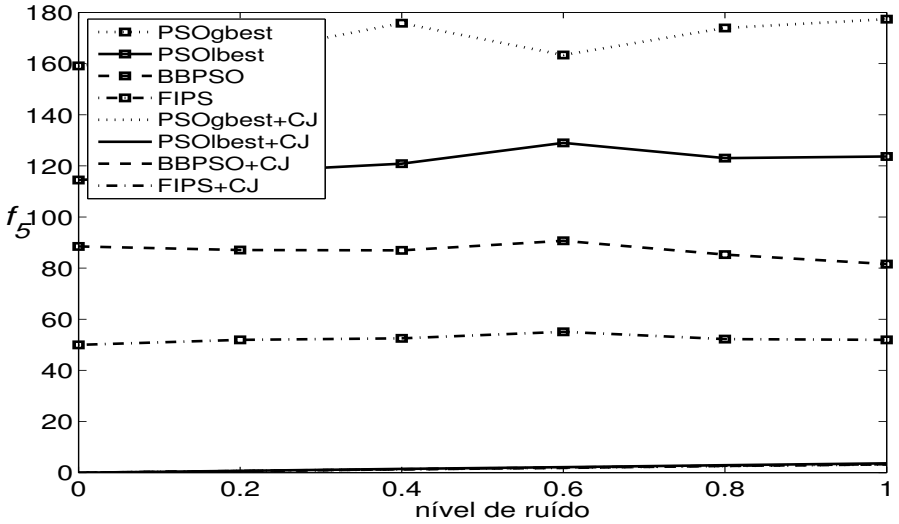


Figura 6.15: Variação da qualidade da solução para a função Rastrigin para ambas as versões dos algoritmos populacionais, original e híbrida com saltos caóticos.

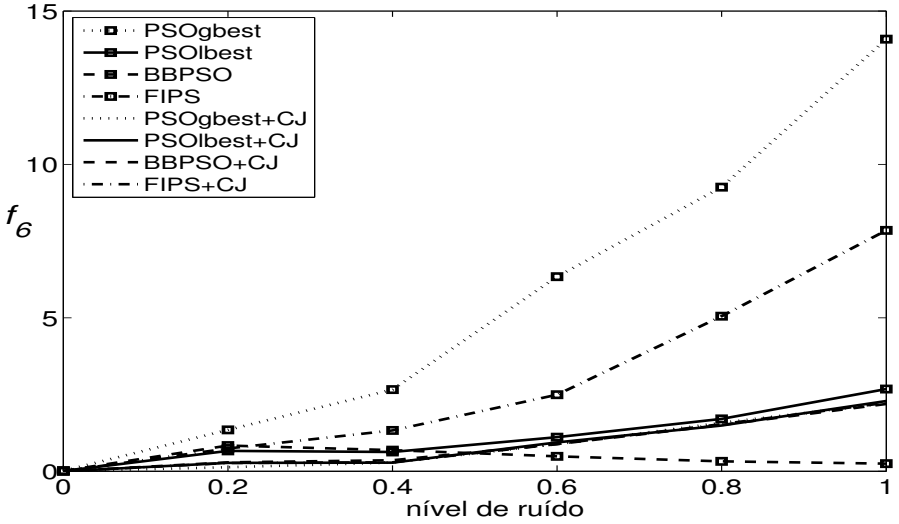
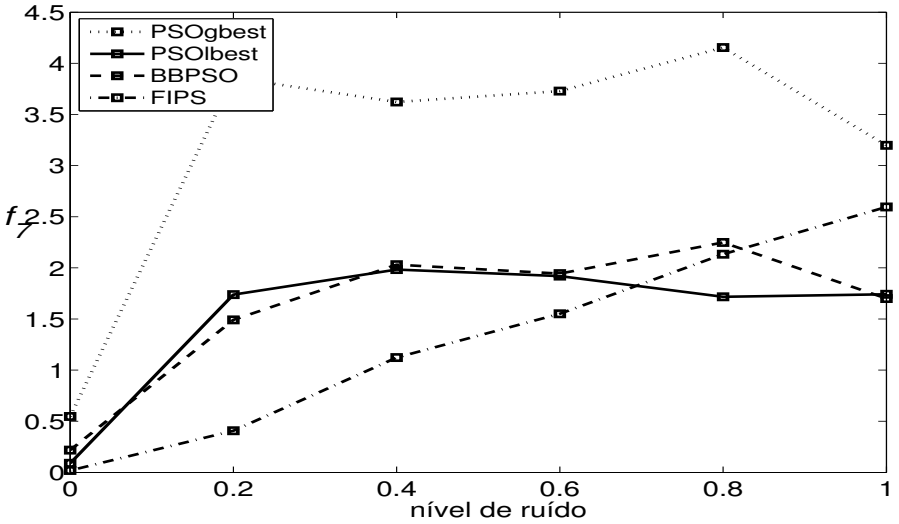
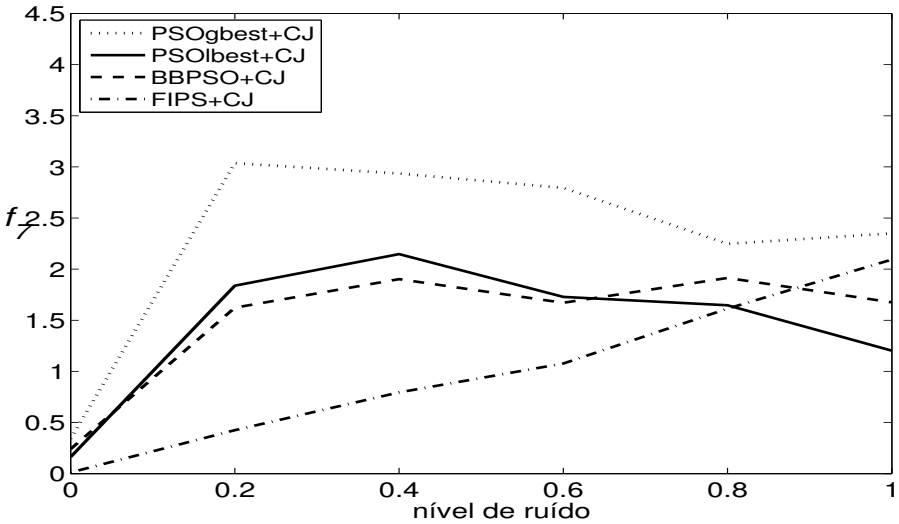


Figura 6.16: Variação da qualidade da solução para a função Griewank para ambas as versões dos algoritmos populacionais, original e híbrida com saltos caóticos.



(a) Algoritmos originais



(b) Abordagens híbridas usando saltos caóticos

Figura 6.17: Variação da qualidade da solução para a função Generalizada Penalizada.

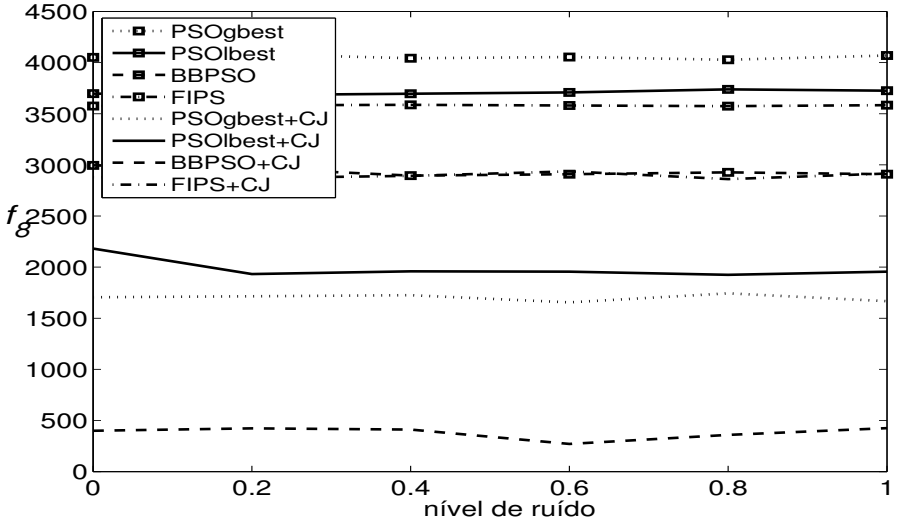


Figura 6.18: Variação da qualidade da solução para a função Schwefel para ambas as versões dos algoritmos populacionais, original e híbrida com saltos caóticos.

6.7.3 Diagrama Box e Whisker

Na estatística descritiva, um *box-plot* (TUKEY, 1977) é uma maneira conveniente de ilustrar graficamente grupos numéricos de dados por meio de cinco medidas estatísticas: a menor observação (mínima amostra), quartil inferior (Q1), mediana (Q2), quartil superior (Q3) e maior observação (amostra máxima). Os *box-plots* exibem as diferenças entre as populações sem fazer qualquer suposição sobre a distribuição estatística implícita: o método é não paramétrico. A Figura 6.19 contém uma representação gráfica de um *box-plot*. Os espaço entre as diferentes partes do diagrama indicam o grau de dispersão e assimetria nos dados. O topo e a base do *box-plot* são sempre o 25° e o 75° percentil (quartis inferior e superior) respectivamente, e a faixa próxima ao centro é sempre o 50° percentil (a mediana).

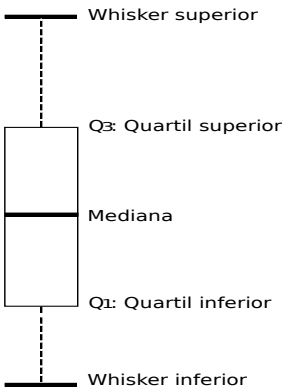


Figura 6.19: Representação gráfica de um *Box-plot*.

Esses valores são considerados estatisticamente robustos. Os *whiskers* marcam os valores mínimos e máximos a menos que excedam $1.5 \cdot IQR$. O IQR é o intervalo inter quartil: a distância entre Q_1 e Q_3. Caso haja observações que estejam além de $1.5 \cdot IQR$ ou até mesmo $3 \cdot IQR$, elas são consideradas valores pouco e muito extremos (*outliers*), respectivamente. O gráfico 6.20 ilustra o conceito de maneira qualitativa (as distâncias não são corretas).

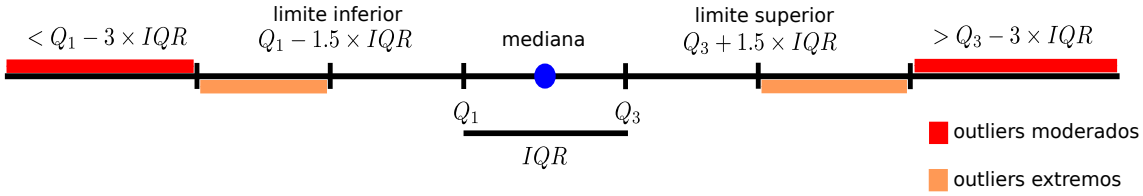
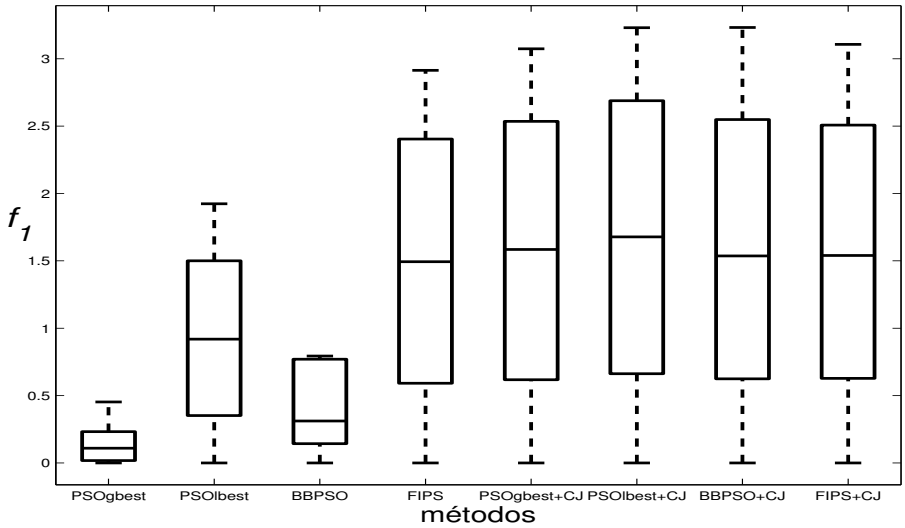


Figura 6.20: Representação qualitativa dos conceitos de um diagrama Box-plot.

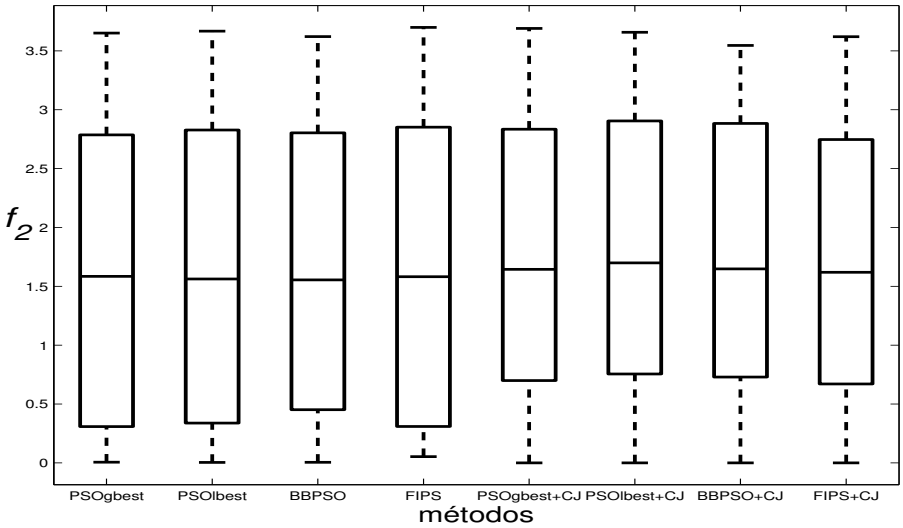
As Figuras 6.21(a) até 6.21(h) representam os *Box-plots* indicando o comportamento dos valores médios dos resultados de 50 execuções obtidos pelos algoritmos originais (PSOgbest, PSOlbest, BBPSO and FIPS) e aperfeiçoados com a técnica de *jump* caótico (PSOgbest+CJ, PSOlbest+CJ, BBPSO+CJ and FIPS+CJ).

Observando-se as Tabelas 6.2, 6.3, 6.4 e 6.5, conclui-se que embora o desempenho dos algoritmos populacionais deteriore com o aumento no nível de ruído, as soluções encontradas pelas versões modificadas são superiores às soluções obtidas pelas versões originais. Isso pode ser melhor visualizado nos *box-plots* das Figuras 6.21(a) até 6.21(h) os quais indicam a eficiência da estratégia de *jump* em ambos os ambientes, estático e ruidoso.

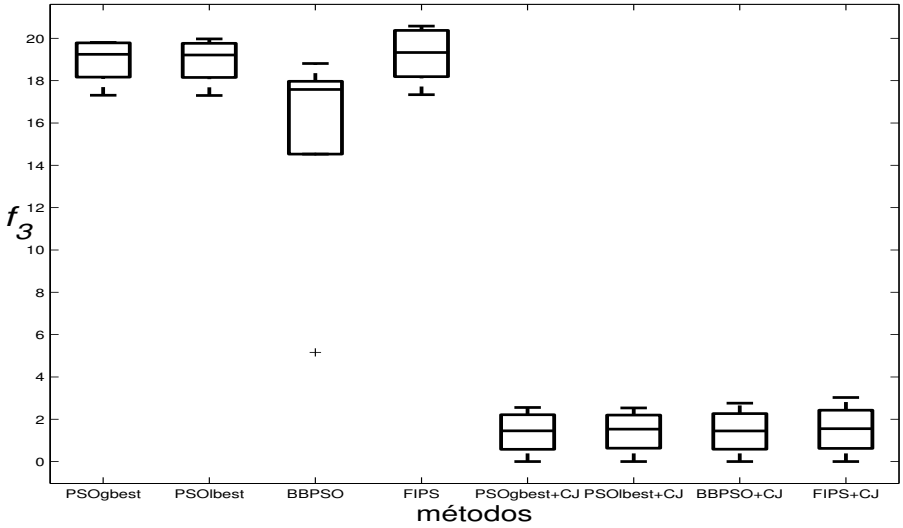


(a) Função Sphere

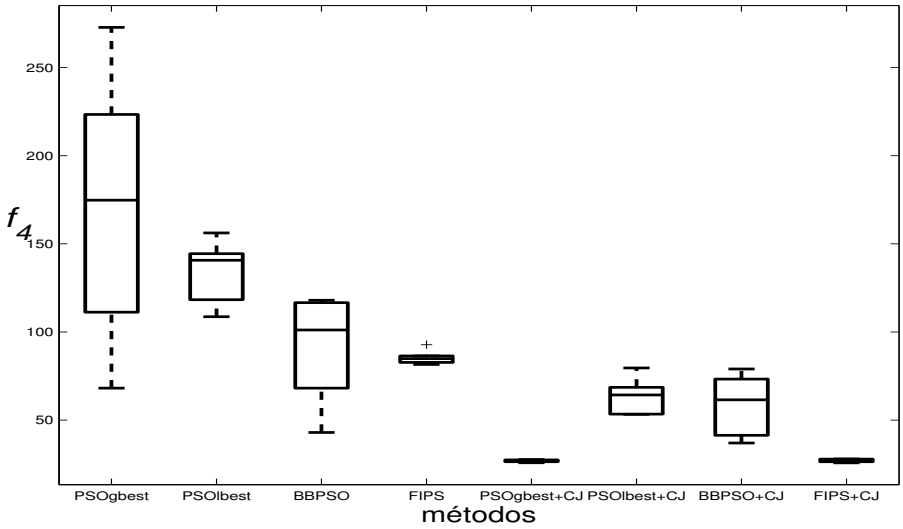
Figura 6.21: Diagramas *Box-plots* para representação da robustez dos algoritmos.



(b) Função Schaffer's F6

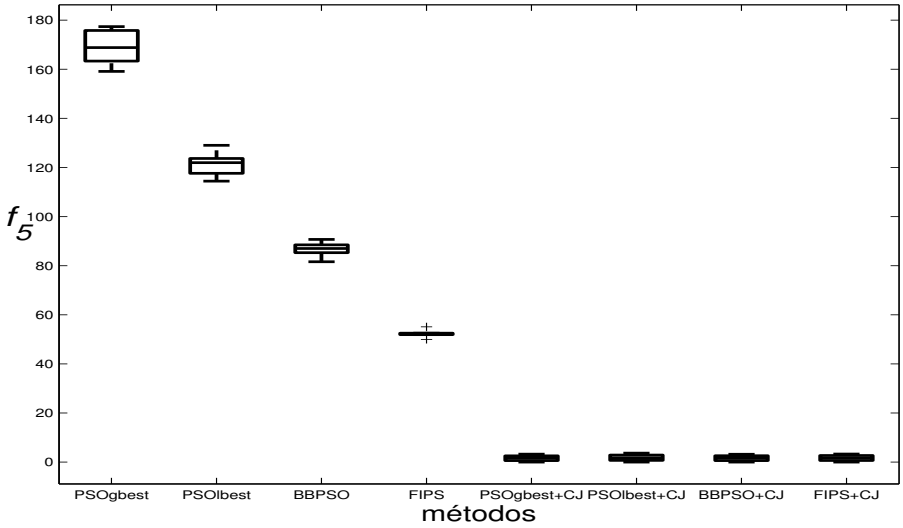


(c) Função Ackley

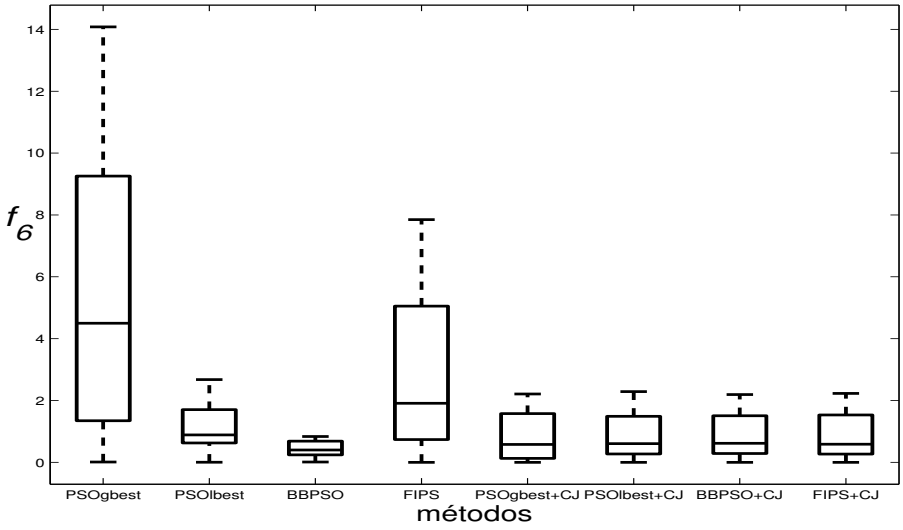


(d) Função Rosenbrock

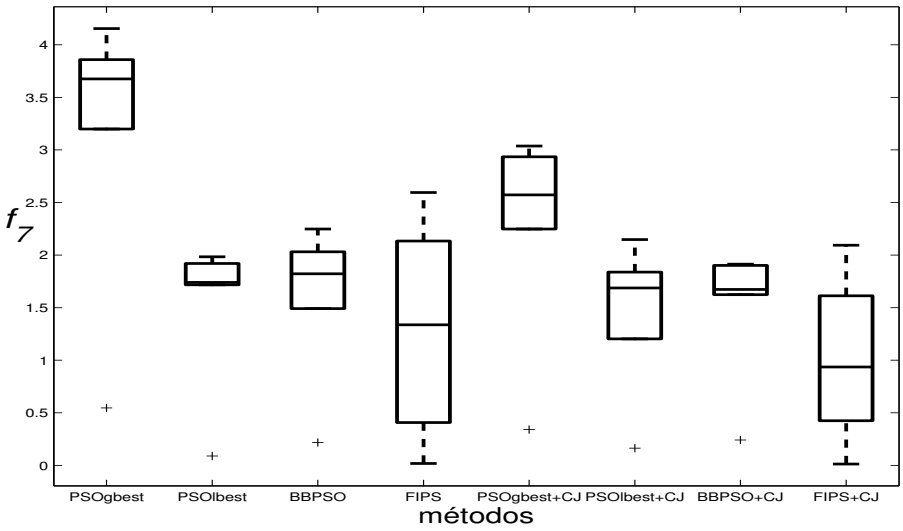
Figura 6.21: (continuação)



(e) Função Rastrigin

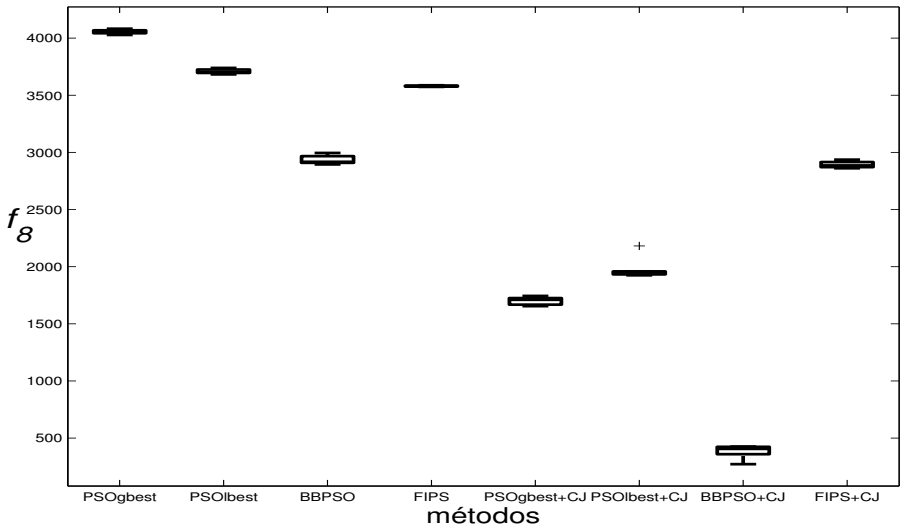


(f) Função Griewank



(g) Função Generalizada Penalizada

Figura 6.21: (continuação)



(h) Função Schwefel

Figura 6.21: (continuação)

7 Conclusão e Trabalhos Futuros

"Through the mystic dawn
Another world, where you come from
If there is a way
Don't hide don't be afraid
Across the sky
The rays of light, on wings you fly
Wherever you are, set your soul free
Won't you come with me
Unlock the doors, set free the dream."

Stronghold
Symphorce

O objetivo principal deste trabalho é a realização de um estudo investigativo experimental de algoritmos populacionais em ambientes ruidosos. Além disso, foi analisado também o comportamento de uma estratégia recém proposta, denominada de estratégia de *jump*, em ambientes incertos. Quatro versões do algoritmo *Particle Swarm Optimization* (PSO) foram adotadas nos experimentos. São elas: PSO canônico (topologia global), PSO padrão (topologia local), *Bare Bones* (BBPSO) e *Fully Informed Particle Swarm* (FIPS). Os experimentos foram realizados principalmente com um conjunto de funções multimodais com muitos mínimos locais. Pode-se notar por meio dos gráficos que o ruído modifica seriamente a paisagem de várias funções, proporcionando um aumento de dificuldade no processo de otimização em virtude da inclusão de muitos falsos mínimos.

O PSO se tornou uma heurística comum na comunidade de otimização. As duas versões do PSO, com modelo global e local, foram adotadas neste trabalho com propósito de servir de referência para comparação do desempenho de novas técnicas propostas. O BBPSO foi investigado porque este é um algoritmo simples, bastante

efetivo e, apesar de usar informação global da população, ele amostra o espaço de possível soluções. A razão do algoritmo FIPS ter sido investigado é que ele aparenta possuir uma característica interessante no caso de problemas ruidosos. Em virtude das partículas utilizarem informações provenientes de todos os seus vizinhos, e não apenas do melhor indivíduo presente na vizinhança, este método poderia atenuar a influência do ruído, funcionando como uma espécie de abordagem embutida de re-avaliação das funções objetivos. Ainda não havia sido reportado nenhum trabalho considerando a versão FIPS em ambientes ruidosos, em que a paisagem da função é corrompida por ruído.

Na primeira parte deste trabalho, foi investigada a abordagem combinando os algoritmos populacionais com a estratégia de *jump* para escapar de mínimos locais. Baseado no fato de que as partículas não necessitam seguir sempre o mesmo regime, estratégia de saltos simples é considerada uma abordagem com dois regimes dinâmicos, para evitar a convergência prematura do algoritmo original. Seu objetivo é permitir que as partículas da população "saltem" pelo espaço de busca, sempre que necessário, na tentativa de escapar de mínimos locais. Essa estratégia é usada somente quando há indícios de estagnação das partículas, ou seja, quando não ocorrem melhorias no valor da função objetivo. Quando há melhoria da *fitness*, as partículas da população movem-se pelo espaço de busca segundo a estratégia tradicional do PSO, FIPS ou BBPSO. A estratégia de *jump* foi inicialmente apresentada baseada nas distribuições de probabilidade Gaussiana e de Cauchy. Resultados muito bons foram obtidos. Eles sugerem que todas as quatro versões dos algoritmos populacionais melhoraram bastante seus desempenhos ao incorporarem essa estratégia, principalmente em problemas multimodais.

Posteriormente, com base nos resultados promissores da estratégia de *jump*, foi investigado o uso de sequências caóticas ao invés de números randômicos gerados anteriormente segundo distribuições de probabilidade. Matematicamente, sistemas não-lineares apresentam um comportamento determinístico não periódico e são muito sensíveis às condições iniciais, isto é, pequenas pertubações nas condições iniciais do sistema dinâmico provocam enormes variações. Especificamente, o mapa caótico considerado nos experimentos ruidosos foi o mapa logístico. Em consequência do mapa caótico ser utilizado na dinâmica de dois regimes exatamente quando as partículas estão estagnadas ou presas em um mínimo local, a característica de gerar números próximos aos dois extremos do intervalo [0; 1], é interessante para gerar novos pontos no espaço de busca afastados da região na qual não ocorrem melhorias da função

objetivo. Não obstante a existência de diferentes mecanismos de busca caótica na literatura especializada sobre algoritmos sócio-bio-inspirados, a ideia de uma estratégia híbrida de jumps caóticos e determinísticos pode ser considerada original. A sequência caótica aqui não é usada dentro do PSO. Ao invés disso, adota-se a sequência caótica somente quando nenhuma melhoria é detectada. Em ambientes ruidosos, a ideia subjacente à abordagem caótica de dois regimes é importante pois permite que as partículas escapem de falsos atratores criados devido à inclusão de ruído.

A estratégia de jump adicionada aos algoritmos populacionais demonstrou ser eficaz quando aplicada tanto às funções estáticas quanto às funções ruidosas pois aumentam as chances das partículas escaparem de mínimos locais. A implementação dessa abordagem é bastante simples e não eleva os custos computacionais. Além disso, o uso de um gerador de sequências caóticas como alternativa às distribuições de probabilidade é responsável por manter a abordagem eficiente e efetiva.

Um estudo de sensitividade foi conduzido utilizando-se o PSO canônico. Análise sensitiva é uma técnica muito utilizada para alterar sistematicamente os parâmetros de um modelo e determinar os efeitos dessas alterações. A compreensão de como o modelo responde às variações em seus parâmetros de entrada é de fundamental importância para assegurar seu uso correto. Os resultados obtidos neste trabalho podem ser melhorados com ajustes finos dos parâmetros necessários à abordagem. Os valores utilizados foram escolhidos de modo genérico e abrangente com o único intuito de servir como um bom ponto de partida para pesquisadores interessados no uso do algoritmo híbrido proposto; são apenas recomendações iniciais. Os melhores valores continuam sendo dependentes de cada problema, e necessitam de ajuste fino para cada problema. Um trabalho futuro interessante é a investigação da abordagem aqui apresentada sem a necessidade de parâmetros.

Introduzir dinamismo na topologia do FIPS pode ser uma direção interessante e atraente para futuras pesquisas na área de ambientes dinâmicos. O uso de vizinhança dinâmica em ambientes ruidosos pode auxiliar na manutenção da diversidade populacional, enquanto evita a tendência dos indivíduos de se moverem rapidamente em direção a um falso ótimo induzido pelo ruído. A despeito de boas expectativas, ainda não foi possível alcançar bons resultados seguindo uma metodologia dinâmica. Contudo, acredita-se que introduzir dinamismo na topologia do FIPS pode ser uma direção interessante e atraente para futuras pesquisas na área de ambientes dinâmicos.

No contexto de otimização ruidosa, um algoritmo para ser considerado robusto precisa alcançar soluções satisfatórias sistematicamente. Os resultados experimentais sustentam a hipótese de que a estratégia de *jump*, além de obter bons resultados sob baixos níveis de ruído, também apresentam menor deterioração do desempenho, à proporção que o nível de ruído aumenta, comparada com os métodos em suas versões originais.

A estratégia investigada neste trabalho pode ser facilmente combinada com outros algoritmos evolutivos. Trabalhos futuros poderiam analisar outras abordagens hibridizadas com *jump* em ambientes ruidosos.

Outro ponto interessante para análises futuras é o estudo profundo da abordagem para descobrir regras gerais e implicações teóricas subjacentes ao algoritmo apresentado. Além disso, ainda é necessária a realização de um estudo detalhado sobre as características e peculiaridades de cada função testada para concluir, especificamente, em quais classes de problemas multimodais a metodologia investigada é mais atraente.

Referências Bibliográficas

AKAT, S. B.; GAZI, V. Particle swarm optimization with dynamic neighborhood topology: three neighborhood strategies and preliminary results. *IEEE Swarm Intelligence Symposium*, p. 1–8, 2008.

ALATAS, B.; AKIN, E.; OZER, A. B. Chaos embedded particle swarm optimization algorithms. *Chaos, Solitons & Fractals*, v. 40, n. 4, p. 1715–1734, 2009.

ANGELINE, P. J. Using selection to improve particle swarm optimization. In: IEEE. *International Conference on Evolutionary Computation*. [S.l.], 1998. p. 84–89.

BARAS, J. S.; TAN, X.; HOVARESHTI, P. Decentralized control of autonomous vehicles. In: IEEE. *Proceedings of the IEEE Conference on Decision and Control*. [S.l.], 2004. v. 2, p. 1532–1537.

BENI, G.; WANG, J. Swarm intelligence in cellular robotic systems. *Proceedings of NATO Advanced Workshop on Robots and Biological Systems*, v. 102, p. 703–703, 1989.

BERGH, F. Van den. *An analysis of particle swarm optimizers*. Tese (Doutorado) — Ph.D. thesis, Department of Computer Science, University of Pretoria, 2002.

BERGH, F. Van den; ENGELBRECHT, A. P. Training product unit networks using cooperative particle swarm optimisers. In: IEEE. *International Joint Conference on Neural Networks*. [S.l.], 2002. v. 1, p. 126–131.

BERGH, F. Van den; ENGELBRECHT, A. P. A study of particle swarm optimization particle trajectories. *Information Sciences*, Elsevier, v. 176, n. 8, p. 937–971, 2006.

BONABEAU, E.; DORIGO, M.; THERAULAZ, G. *Swarm intelligence: from natural to artificial systems*. [S.l.]: Oxford University Press, USA, 1999.

BOURJOT, C.; CHEVRIER, V.; THOMAS, V. A new swarm mechanism based on social spiders colonies: from web weaving to region detection. *Web Intelligence and Agent Systems*, v. 1, n. 1, p. 47–64, 2003.

BRATTON, D.; KENNEDY, J. Defining a standard for particle swarm optimization. *Proceedings of the IEEE Swarm Intelligence Symposium*, p. 120–127, 2007.

CAMPBELL, D. T. Blind variation and selective retention in creative thought asin other knowledge processes. *Psychological Review*, v. 67, p. 380–400, 1960.

CAPONETTO, R. et al. Chaotic sequences to improve the performance of evolutionary algorithms. *IEEE Transactions on Evolutionary computation*, v. 7, n. 3, p. 289–304, 2003.

CARLISLE, A. *Applying the particle swarm optimizer to non-stationary environments*. [S.l.]: Auburn University, 2002.

CARLISLE, A.; DOZIER, G. An off-the-shelf PSO. In: *Proceedings of the workshop on particle swarm optimization*. [S.l.: s.n.], 2001. v. 1, p. 1–6.

CARO, G. D.; DORIGO, M. Ant colonies for adaptive routing in packet-switched communications networks. In: *Proceedings of the International Conference on Parallel Problem Solving from Nature*. [S.l.]: Springer-Verlag, 1998. p. 673–682.

CASTRO, L. N.; TIMMIS, J. *Artificial immune systems: a new computational intelligence approach*. [S.l.]: Springer Verlag, 2002.

CHUANWEN, J.; BOMPARD, E. A self-adaptive chaotic particle swarm algorithm for short term hydroelectric system scheduling in deregulated environment. *Energy Conversion and Management*, v. 46, n. 17, p. 2689–2696, 2005.

CLERC, M.; KENNEDY, J. The particle swarm – explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, v. 6, n. 1, p. 58–73, 2002.

COELHO, L. S.; HERRERA, B. M. Fuzzy identification based on a chaotic particle swarm optimization approach applied to a nonlinear yo-yo motion system. *IEEE Transactions on Industrial Electronics*, v. 54, n. 6, p. 3234–3245, 2007.

COELHO, L. S.; KROHLING, R. A. Predictive controller tuning using modified particle swarm optimization based on cauchy and gaussian distributions. *Soft computing: methodologies and applications*, Springer, p. 287–298, 2005.

COELHO, L. S.; MARIANI, V. C. Use of chaotic sequences in a biologically inspired algorithm for engineering design optimization. *Expert Systems with Applications*, v. 34, n. 3, p. 1905–1913, 2008.

DENEUBOURG, J. L. et al. The self-organizing exploratory pattern of the argentine ant. *Journal of Insect Behavior*, Springer, v. 3, n. 2, p. 159–168, 1990.

DORIGO, M.; BIRATTARI, M.; STÜTZLE, T. Ant colony optimization. *IEEE Computational Intelligence Magazine*, v. 1, n. 4, p. 28–39, 2006.

DORIGO, M.; CARO, G. D.; GAMBARDELLA, L. M. Ant algorithms for discrete optimization. *Artificial life*, MIT Press, v. 5, n. 2, p. 137–172, 1999.

DORIGO, M.; GAMBARDELLA, L. M. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, IEEE, v. 1, n. 1, p. 53–66, 2002.

DORIGO, M.; MANIEZZO, V.; COLORNI, A. *Positive feedback as a search strategy*. [S.l.], 1991.

EBERHART, R.; KENNEDY, J. A new optimizer using particle swarm theory. In: IEEE. *Proceedings of the International Symposium on Micro Machine and Human Science*. [S.l.], 1995. p. 39–43.

EBERHART, R.; SHI, Y. Comparing inertia weights and constriction factors in particle swarm optimization. In: IEEE. *Proceedings of the Congress on Evolutionary Computation*. [S.l.], 2002. v. 1, p. 84–88.

EBERHART, R. C.; SHI, Y.; KENNEDY, J. *Swarm intelligence*. [S.l.]: Elsevier, 2001.

GEHLHAAR, D. K.; FOGEL, D. B. Tuning evolutionary programming for conformationally flexible molecular docking. In: *Conference on Evolutionary Programming*. [S.l.: s.n.], 1996. p. 419–429.

GLEICK, J.; GLAZIER, J.; GUNARATNE, G. Chaos: Making a new science. *Physics Today*, v. 41, p. 79, 1988.

GOLDBERGER, A. L.; RIGNEY, D. R.; WEST, B. J. Chaos and fractals in human physiology. *Scientific American*, v. 262, n. 2, p. 42–49, 1990.

GRANOVETTER, M. S. The strength of weak ties. *American journal of sociology*, v. 78, n. 6, p. 1360–1380, 1973.

HU, X.; EBERHART, R. Multiobjective optimization using dynamic neighborhood particle swarm optimization. *Proceedings of the Evolutionary Computation*, p. 1677–1681, 2002.

JORDAN, J.; HELWIG, S.; WANKA, R. Social interaction in particle swarm optimization, the ranked FIPS, and adaptive multi-swarms. In: *Proceedings of the conference on Genetic and evolutionary computation*. [S.l.: s.n.], 2008. p. 49–56.

KENNEDY, J. The behavior of particles. In: *Proceedings of the Evolutionary Programming Conference*. [S.l.: s.n.], 1998. p. 579–589.

KENNEDY, J. Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance. In: *Proceedings of the Congress on Evolutionary Computation*. [S.l.: s.n.], 1999. v. 3, p. 1931–1938.

KENNEDY, J. Bare bones particle swarms. *Proceedings of the IEEE Swarm Intelligence Symposium*, p. 80–87, 2003.

KENNEDY, J. Probability and dynamics in the particle swarm. In: IEEE. *Congress on Evolutionary Computation*. [S.l.], 2004. p. 340–347.

KENNEDY, J. Dynamic-probabilistic particle swarms. In: *Proceedings of the Conference on Genetic and Evolutionary Computation*. [S.l.: s.n.], 2005. p. 201–207.

KENNEDY, J. Why does it need velocity? In: IEEE. *Proceedings IEEE Swarm Intelligence Symposium*. [S.l.], 2005. p. 38–44.

KENNEDY, J. How it works: Collaborative trial and error. *International Journal of Computational Intelligence Research*, Research India Publications, v. 4, p. 71–78, 2008.

KENNEDY, J.; EBERHART, R. Particle swarm optimization. *Proceedings of the IEEE International Conference on Neural Networks*, p. 1941–1948, 1995.

KENNEDY, J.; MENDES, R. Topological structure and particle swarm performance. In: *Proceedings of the Congress on Evolutionary Computation*. [S.l.: s.n.], 2002. p. 1671–1676.

KENNEDY, J.; MENDES, R. Neighborhood topologies in fully informed and best-of-neighborhood particle swarms. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, v. 36, n. 4, p. 515–519, 2006.

KORB, O.; STÜTZLE, T.; EXNER, T. Application of ant colony optimization to structure-based drug design. *Ant Colony Optimization and Swarm Intelligence*, Springer, p. 247–258, 2006.

KROHLING, R. A. Gaussian swarm: a novel particle swarm optimization algorithm. *IEEE Conference on Cybernetics and Intelligent Systems*, p. 372–376, 2004.

KROHLING, R. A. Gaussian particle swarm with jumps. *IEEE Congress on Evolutionary Computation*, v. 2, p. 1226–1231, 2005.

KROHLING, R. A.; COELHO, L. S. Coevolutionary particle swarm optimization using Gaussian distribution for solving constrained optimization problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, IEEE, v. 36, n. 6, p. 1407–1416, 2006.

KROHLING, R. A.; MENDEL, E. Bare bones particle swarm with Gaussian or Cauchy jumps. *IEEE Congress on Evolutionary Computation*, Trondheim, Norway, p. 3285–3291, 2009.

KROHLING, R. A.; MENDEL, E.; CAMPOS, M. Swarm algorithms with chaotic jumps for optimization of multimodal functions. *Engineering Optimization*, 2011.

LASOTA, A.; MACKEY, M. *Chaos, fractals, and noise: stochastic aspects of dynamics*. New York: Springer Verlag, 1994.

LATANE, B. The psychology of social impact. *American Psychologist*, v. 36, n. 4, p. 343–356, 1981.

LEE, C. Y.; YAO, X. Evolutionary programming using mutations based on the Lévy probability distribution. *IEEE Transactions on Evolutionary Computation*, v. 8, n. 1, p. 1–13, 2004.

LIU, B. et al. Improved particle swarm optimization combined with chaos. *Chaos, Solitons & Fractals*, v. 25, n. 5, p. 1261–1271, 2005.

MENDEL, E.; KROHLING, R. A.; CAMPOS, M. Swarm Algorithms with Chaotic Jumps Applied to Noisy Optimization Problems. *Information Sciences*, Elsevier, 2010.

MENDES, R. *Population topologies and their influence in particle swarm performance*. Tese (Doutorado), 2004.

MENDES, R.; KENNEDY, J.; NEVES, J. Avoiding the pitfalls of local optima: how topologies can save the day. IEEE, 2003.

MENDES, R.; KENNEDY, J.; NEVES, J. The fully informed particle swarm: simpler, maybe better. *IEEE Transactions on Evolutionary Computation*, v. 8, n. 3, p. 204–210, 2004.

MENDES, R.; NEVES, J. What makes a successful society?: Experiments with population topologies in particle swarms. *Lecture notes in computer science*, Springer, p. 346–355, 2004.

MENG, H. J. et al. A hybrid particle swarm algorithm with embedded chaotic search. In: IEEE. *Conference on Cybernetics and Intelligent Systems*. [S.l.], 2005. v. 1, p. 367–371.

MOHAIS, A. et al. Neighborhood re-estructuring in particle swarm optimization. *Australian Conference on Artificial Intelligence*, p. 776–785, 2005.

OHNO, S. Codon preference is but an illusion created by the construction principle of coding sequences. *Proceedings of the National Academy of Sciences of the United States of America*, v. 85, n. 12, p. 4378, 1988.

PAN, F. et al. An analysis of bare bones particle swarm. In: *IEEE Swarm Intelligence Symposium*. [S.l.: s.n.], 2008. p. 21–23.

PASSINO, K. M. Biomimicry of bacterial foraging for distributed optimization and control. *Control Systems Magazine*, IEEE, v. 22, n. 3, p. 52–67, 2002.

PITCHER, T. J.; PARTRIDGE, B. L.; WARDLE, C. S. A blind fish can school. *Science*, AAAS, v. 194, n. 4268, 1976.

POLI, R.; LANGDON, W. B. Markov chain models of bare-bones particle swarm optimizers. In: *Proceedings of the Conference on Genetic and Evolutionary Computation*. [S.l.: s.n.], 2007. p. 142–149.

POTTS, W. K. The chorus-line hypothesis of manoeuvre coordination in avian flocks. Nature, v. 309, p. 344–345, 1984.

REYNOLDS, C. W. Flocks, herds and schools: A distributed behavioral model. In: *Proceedings of the conference on Computer graphics and interactive techniques*. [S.l.]: ACM, 1987. p. 25–34.

RUDOLPH, G. Self-adaptive mutations may lead to premature convergence. *IEEE Transactions on Evolutionary Computation*, v. 5, n. 4, p. 410–414, 2001.

SHAW, E. The schooling of fishes. *Scientific American*, v. 206, p. 128–138, 1962.

SHI, Y.; EBERHART, R. Modified particle swarm optimizer. In: *The IEEE International Conference on Evolutionary Computation*. [S.l.: s.n.], 1998. p. 69–73.

SHI, Y.; EBERHART, R. Parameter selection in particle swarm optimization. In: SPRINGER. *Evolutionary Programming VII*. [S.l.], 1998. p. 591–600.

SUGANTHAN, P. N. Particle swarm optimiser with neighbourhood operator. *IEEE Congress on Evolutionary Computation*, p. 1945–1950, 1999.

TUKEY, J. W. Exploratory data analysis. 1977.

VICTOIRE, T.; JEYAKUMAR, A. E. Hybrid PSO-SQP for economic dispatch with valve-point effect. *Electric Power Systems Research*, Elsevier, v. 71, n. 1, p. 51–59, 2004.

WESSON, R. *Beyond natural selection*. [S.l.]: The MIT Press, 1993.

XIANG, T.; LIAO, X.; WONG, K. An improved particle swarm optimization algorithm combined with piecewise linear chaotic map. *Applied Mathematics and Computation*, v. 190, n. 2, p. 1637–1645, 2007.

YANG, D.; LI, G.; CHENG, G. On the efficiency of chaos optimization algorithms for global optimization. *Chaos, Solitons & Fractals*, v. 34, n. 4, p. 1366–1375, 2007.

YANG, K.; NOMURA, H. Quantum-Behaved Particle Swarm Optimization with Chaotic Search. *IEICE Transactions on Information and Systems*, v. 91, n. 7, p. 1963–1970, 2008.

YAO, X.; LIU, Y. Fast evolutionary programming. In: *Proceedings of the Fifth Annual Conference on Evolutionary Programming*. [S.l.: s.n.], 1996. p. 451–460.

YAO, X.; LIU, Y.; LIN, G. Evolutionary programming made faster. *Evolutionary Computation, IEEE Transactions on*, IEEE, v. 3, n. 2, p. 82–102, 2002.